
ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK
MP
Grund
Heft 1 · 1987
Mikroprozessortechnik
VEB Verlag Technik Berlin
ISSN 0233-2892
Programmierung in C
16-Bit-Mehrmikrorechnersystem
Kleincomputer KC 87

vorgestellt:

Kleinrechnersystem SM 52/12 (ČSSR)

Technische Daten:

Prozessor: mikroprogrammgesteuert, 99-Bit-Steuerwort, Ausführungsdauer des Mikrobefehls: 200 ns, Größe des Mikroprogramm-Steuerspeichers: 4 K Wörter/Wort 99 Bits PROM/1 K Wörter VOCS/RAM/ Innere Daten-Sammelleitung: 32 Bits, CACHE-Speicher 8 K Bytes, Zweiwegspeicher, Zugriffzeit beim Lesen von CACHE-Speicher 290 ns

Befehlsvorrat:
16 × 32-Bit-Universalregister
248 Basisbefehle
32 Prioritätsunterbrechungsebenen
Datentypen:
Ganze Zahlen, Gleitkomma, Zeichenketten, Bitfelder verschiedener Länge und numerische Ketten
Virtueller Adressenbereich 4 GByte
Physischer Adressenbereich 1 GByte
Adressenmodi 9

Operationsspeicher:
physische Speichergröße
8 MByte /RAM 16 K × 1/ mit zwei Steuereinheiten
32 MByte /RAM 64 K × 1/
Parität
8 Bits ECC auf 64-Bit-Vierwort
Technologie
16 K × 1 Bit dynamischer RAM
Lesezyklus
800 ns 64 Bits
Zugriffszeit
800 ns /Lesen 64 Bits/
Schreibzyklus
1000 ns /64 Bits/
Adapter der gemeinsamen Sammelleitung
max. 4 Stck.
max. Übertragungsgeschwindigkeit
1,5 M Bits/s
Datenwege mit Puffer
15 max. 8-Bytes-Puffer in jedem Weg
max. Belegungszahl für die Sammelleitung
15 einschl. Übertrager
Unterbrechung
direkt, vektoriell mittels ASZ

programmierbarer Kunden-Mikroprogrammspeicher:
Größe
1 K Wort/99 Bits pro Wort/
Technologie
Schreibspeicher RAM, 70 ns Zugriffszeit

Abmessungen:

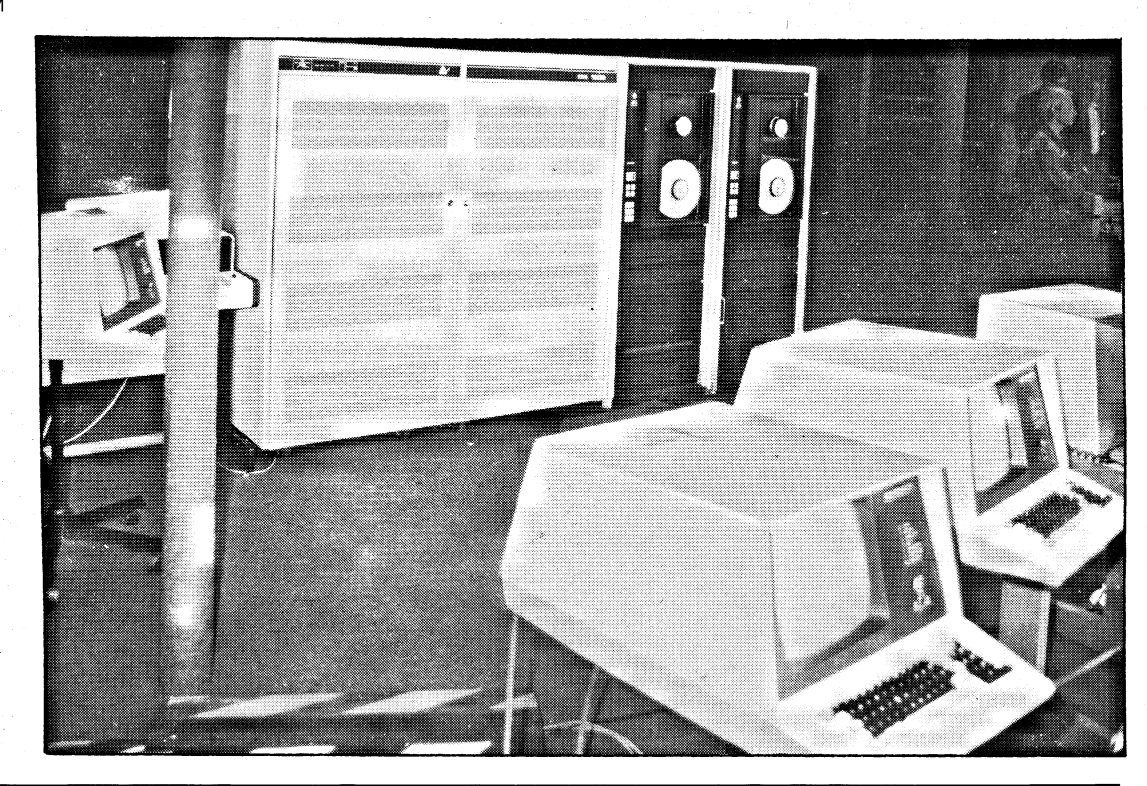
Höhe	1600 mm
Breite	1600 mm
Tiefe	800 mm

Erstmals zur Maschinenbaumesse Brno 1985 und auch in diesem Jahr wurde von KOVO der erste 32-Bit-Rechner des SKR, die Anlage SM 52/12, vorgestellt, die gegenüber bisherigen 16-Bit-Anlagen eine Reihe von Vorteilen hat. (Die folgenden Angaben entnahmen wir einem Prospekt von datasystem.)

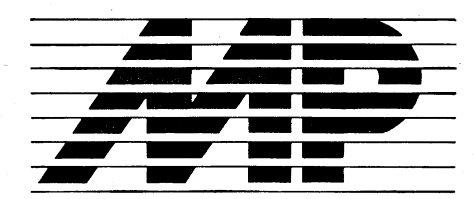
Charakteristik

Das *Prozessorwort* hat 32 Bits und ermöglicht die Adressierung eines 4 GByte großen virtuellen Speichers, aufgeteilt auf je 2 GByte Operationssystem und Verarbeitungssystem.
Der *Operativspeicher* hat 2 MByte und ist auf 8 MByte erweiterbar.
Es gibt 32 *Unterbrechungspositionen*, davon 16 hardwaregenerierte und 16 Software-Positionen.
Der *Befehlsvorrat* besteht aus 248 Grundbefehlen – zum Beispiel für die Gleitkommaoperationen, die Arbeit mit Zeichenketten und gepackten Dezimalketten – und einem Befehlsvorrat für die Verarbeitung von 16-Bit-Kundenprogrammen.
Der 8-Byte-*Befehlspuffer* ermöglicht es der CPU, während der Abarbeitung eines Befehls den nachfolgenden bereits zu lesen und zu dekodieren und damit die Leistung des Prozessors wesentlich zu erhöhen.
Neben diesem Befehlspuffer als CACHE-System gibt es ein weiteres Register für 128 visuell-physische Übersetzungen von Seitenadressen sowie den eigentlichen *CACHE-Speicher*. Dieser ist ein 8-KByte-Zweiweg-Assoziativspeicher für häufig benutzte Adressen und Befehle. Die Zugriffzeit zum Speicher verringert sich für die Zentraleinheit damit beim Lesen von 1800 ns auf 290 ns.
Die Interaktion zwischen den Hauptfunktionsblöcken wird über folgende Sammelleitungen realisiert: ID-Bus, V-Bus, CS-Bus, UPC, MD-Bus und PA-Bus.
Für das Programmieren in *Assembler* stehen 16 32-Bit-Register zur Verfügung, die bei der Arbeit mit dem ursprünglichen Befehlsvorrat verwendet werden können.
Das *Diagnosesystem* der Anlage besteht aus folgenden Elementen: Operationssystem, Diagnostiksupervisor, Diagnostikprogramme, Position Konsole.
Als *Konsolenuntersystem* dient der Minirechner SM 50/50 mit 16-KWort-Speicher, Bildschirm, Disketten-Laufwerk und Mosaikdrucker.
Auf einem virtuellen Speicher baut das universelle *Betriebssystem* MOS auf, eine Weiterentwicklung des DOS RV. Im Entwicklungsstadium ist das System INMOS. Getestet wird bereits die Kommunikationssoftware SYRPOS. BASIC 32, FORTRAN 77, PASCAL und C sind zur Zeit als Sprachen verfügbar, in Zukunft voraussichtlich u. a. auch PL/1, PROLOG und COBOL.

Foto: Paszkowsky



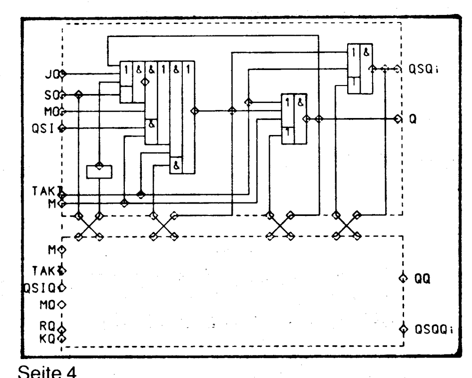
Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0233-2892



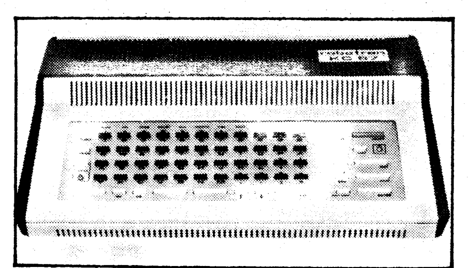
Mikroprozessortechnik, Heft 1 · 87

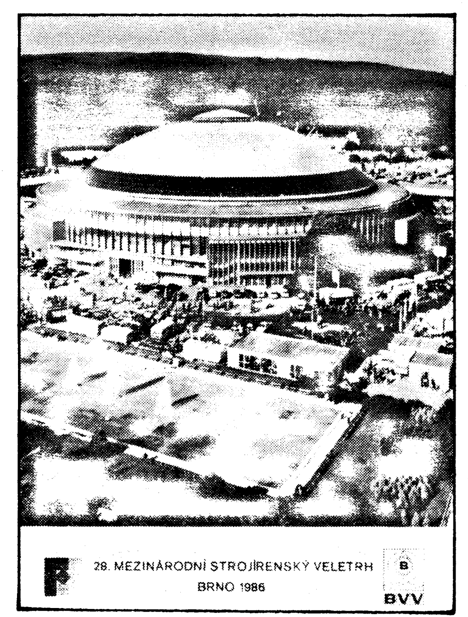
Herausgeber Kammer der Technik, Fachverband Elektrotechnik
Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR-1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd
Verlagsdirektor Klaus Hieronimus
Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat (Tel.: 2870381)
Gestaltung Christina Kaminski (Tel.: 2870288)
Titelbild Tatjana Stephanowitz
Beirat Dr. Ludwig Claßen, Prof. Dr. sc. Dietrich Eckhardt, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Siegmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Jürgen Zaremba
Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik
Gesamtherstellung Druckerei Märkische Volksstimme Potsdam
Erfüllungsort und Gerichtsstand Berlin Mitte.

Redaktionsschluß: 14. November 1986
AN (EDV) 49837
Erscheinungsweise monatlich 1 Heft
Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.
Bezugsmöglichkeiten
DDR: sämtliche Postämter; **SVR Albanien:** Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; **VR Bulgarien:** Direkzia R.E.P., 11a, Rue Paris, Sofia; **VR China:** China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; **ČSSR:** PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; **SFR Jugoslawien:** Jugoslovenska Knjiga, Terazije 27, Beograd; Izdavacko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; **Koreanische DVR:** CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; **Republik Kuba:** Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; **VR Polen:** C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; **SR Rumänien:** D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; **UdSSR:** Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; **Ungarische VR:** P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; **SR Vietnam:** XUNHASABA, 32, Hai Ba Trung, Hanoi; **BRD und Berlin (West):** ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; **Österreich:** Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; **Schweiz:** Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; **Alle anderen Länder:** örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR-7010 Leipzig, und Leipzig Book Service, Talstraße 29, DDR-7010 Leipzig


Seite 4

Seite 19


Seite 22



Seite 28

Fotos: Darre (1), Werkfoto (1)

Inhalt

Informatik-Zentrum des Hochschulwesens der DDR gebildet

Am 4. Oktober 1986 wurde durch den Minister für Hoch- und Fachschulwesen, Prof. Dr. h. c. Böhme, das Informatik-Zentrum des Hochschulwesens der DDR an der Technischen Universität Dresden gegründet. Als Direktor wurde der Rektor der Ingenieurhochschule Dresden, Prof. Dr. rer. oec. Tzschoppe, ernannt.
Das Informatik-Zentrum entsteht aus Kapazitäten der Ingenieurhochschule Dresden und der Technischen Universität Dresden.
Durch den schrittweisen Aufbau einer so zentralisierten Lehr- und Forschungseinrichtung für die Sicherung und Koordinierung des Bildungs- und Forschungsverlaufes im Hochschulwesen der DDR wird der Entwicklung und breiten Anwendung moderner Informations- und Kommunikationstechnologien entsprechend den Bedürfnissen der entwickelten sozialistischen Gesellschaft Rechnung getragen.
Die Ausbildung erfolgt in der Grundstudienrichtung Informatik in den Fachrichtungen

- Theoretische Informatik
- Systemsoftware
- Angewandte Informatik
- Rechnergestaltung und -betrieb.

In der disziplinären Grundlagenforschung und interdisziplinären Nutzanwendung konzentriert sich das Informatik-Zentrum in Einheit von Software und Hardware u. a. auf die Gebiete Rechnerarchitektur, Rechnernetze, Softwaretechnologie, Basissoftware für CAD/CAM und rechnerintegrierte Fertigung (CIM) sowie optische Informationsverarbeitung und Bildverarbeitung.
Die Entwicklung des wissenschaftlichen Nachwuchses und der Ausbau des Systems der Weiterbildung für die Informatik im Hochschulwesen wird in der Wechselwirkung und aktiven Mitarbeit der Kombinate und Betriebe ein wesentliches Aufgabengebiet des Informatik-Zentrums sein. Hierbei gilt es vor allem, neue Formen in gemeinsamer Arbeit zur Erreichung des gestellten Zieles zu finden. Die Bedeutung des Komplexes der Weiterbildung wird durch den Beschluß unterstrichen, am Informatik-Zentrum ein Weiterbildungszentrum Informatik ab 1987 aufzubauen.
Das Informatik-Zentrum wird die gesellschaftlich notwendig bedeutend gewachsenen Anforderungen an Spezialisten der Informatik sowie an die Grundlagen- und Fachausbildung in den technischen, naturwissenschaftlichen, wissenschafts-wissenschaftlichen und weiteren Fachrichtungen auf dem Gebiet der Informatik qualitativ und quantitativ erfüllen helfen.

Prof. Dr. R. Giesecke

Ungarische VR baut Informationssystem aus

Die Beschleunigung des technischen Fortschritts und die breite Einführung von Schlüsseltechnologien in der gesamten Wirtschaft Ungarns wird mit einem Programm angestrebt, das den Ausbau eines Informationssystems unter maximaler Nutzung aller Computer unterschiedlichen Typs vorsieht. Das geplante System soll die Kommunikation zwischen den einzelnen Forschungszentralen, Instituten, Universitäten und Betrieben ermöglichen.
Eine Grundlage für den beschleunigten Informationsaustausch zwischen Forschern wurde vom Zentralen Physikalischen Forschungsinstitut der Ungarischen Akademie der Wissenschaften (KFKI) geschaffen, in dem alle Computer an ein einheitliches System angeschlossen wurden. Das Institut verfügt über sowjetische Rechentechnik vom Typ ES 1040 und ES 1045 sowie über Büro- und Personalcomputer, die nunmehr alle mit dem Rechenzentrum des Instituts verbunden sind.

Kubanische Elektronik-Zeitschrift gegründet

CID heißt die erste kubanische Zeitschrift für Elektronik und Datenverarbeitung. Sie wird vom Nationalen Institut für Automatisierung und Rechentechnik mit einer Auflage von vorerst 5000 Exemplaren herausgegeben.
Die Fachzeitschrift, die computergestützt hergestellt wird, soll sowohl Fachleute als auch interessierte Laien ansprechen.

Spracherkennung

Ein Computer aus dem Laboratorium für „Bionik" der Bulgarischen Akademie der Wissenschaften ist in der Lage, mündliche Kommandos eines Menschen zu verarbeiten. Das gelang unter Ausnutzung zweier Erfindungen zum Erkennen gesprochener Kommandos und von Sprechsignalen. Die bulgarischen Fachleute entwickelten ein Kleinrechnersystem, das zur Steuerung flexibler automatisierter Produktionssysteme, von Robotern und anderen Maschinen vorgesehen ist. Der Operator steuert diese mittels Kleinrechner allein mit seiner Stimme und kann mindestens 20 verschiedene Kommandos erteilen. Sie genügen, um praktisch alle technischen Prozesse in der jeweiligen Produktion auszuführen.
Labortests haben bewiesen, daß der Komplex „menschliche Stimme – Kleinrechner – Mikroprozessor – System einer Maschine" zuverlässig arbeitet. Der Einsatz unter realen Produktionsbedingungen ist vorgesehen.

Das Kombinat VEB Elektro-Apparate-Werke „Friedrich Ebert" zur LFM '87

Die Präsentation in der Halle 15 erfolgt schwerpunktmäßig auf neue Spitzenerzeugnisse der EAW-electronic mit dem modular aufgebauten elektronischen Steuerungs- und Regelungssystem S 2000, das in verschiedenen Ausbaustufen und Konfigurationen für automatisierungstechnische Lösungen der unteren bis mittleren Leistungsklasse gezeigt wird. Dazu gehört das Entwicklungs- und Programmiersystem P 8000, das sowohl für die Programmierung der EAW-electronic S 2000 als auch als universeller Entwicklungsarbeitsplatz in multi-user-Eigenschaft sowie als CAD/CAM-Arbeitsplatz eingesetzt wird. Das Programmier- und Entwicklungssystem für die 8- und 16-Bit-Mikrorechentechnik verfügt über eine Fülle von Software-Komponenten. Dadurch kann der Anwender, aufbauend auf vorhandene erprobte Lösungen, zukunftsorientierte Projekte bei gesicherten Hardware- und Software-Investitionen realisieren.
Die funktionelle Einsatzbreite und die dazugehörige Software wird dem Messebesucher in Form von Problemlösungen angeboten. Dabei sind die Vergabe von Erzeugnis- und Technologie-Lizenzen, die Ausstattung von Produktionsstätten, die Ausbildung von Fachpersonal im Zusammenhang mit der Realisierung von Problemlösungen weitere Schwerpunkte der Angebotstätigkeit.
Das P-8000-System wird in Mikroprozessortechnik 3/1987 ausführlich beschrieben.
Das Sortiment der EAW-electronic wird komplettiert durch modernste Erzeugnisse der Anzeige- und Registriertechnik, wobei besonders der programmierbare Mehrkanalschreiber mit Mikroprozessor PMM 100 und der Grenzwertmelder mit Einchip-Mikrorechner sowie eine breite Palette von Temperaturreglern hervorzuheben sind.

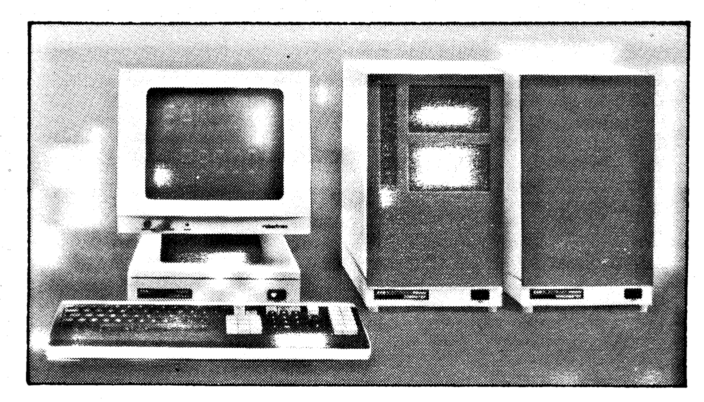


Bild 1 Programmier- und Entwicklungssystem für 8- und 16-Bit-Mikrorechentechnik P 8000 *[Werkfotos (2)]*

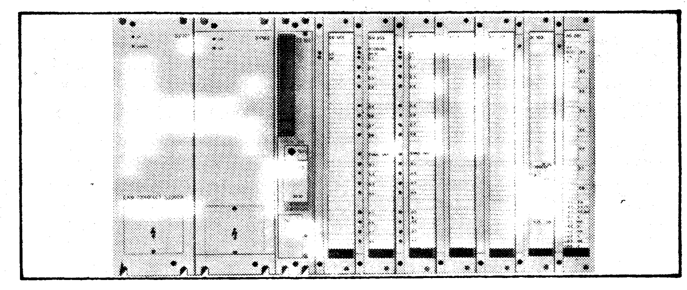


Bild 2 Speicherprogrammierbares Gerätesystem EAW-electronic S 2000

Mikroelektronik – Schlüsseltechnologie für die dynamische Entwicklung unserer Volkswirtschaft

In der Direktive des XI. Parteitages der SED zum Fünfjahrplan für die Entwicklung der Volkswirtschaft der DDR in den Jahren 1986–1990 ist zur Sicherung der dynamischen und effektiven Entwicklung der Produktivkräfte die rasche Entwicklung und umfassende Anwendung der Schlüsseltechnologien, insbesondere der Mikroelektronik, der sich auf ihrer Grundlage entwickelnden modernen Rechentechnik, der rechnergestützten Projektierung, Konstruktion und Produktionsvorbereitung und -steuerung sowie der flexiblen automatisierten Fertigungssysteme festgelegt.

Der Industriebereich Elektrotechnik und Elektronik trägt bei der Verwirklichung dieser Aufgabe sowohl zur Schaffung der materiell-technischen Basis in der Einheit von Soft- und Hardware als auch bei der beispielgebenden Anwendung eine hohe Verantwortung.

Auf der Grundlage der Beschlüsse des XI. Parteitages der SED werden die Kombinate Carl Zeiss JENA und Mikroelektronik Erfurt zu Zentren der Hochtechnologien ausgebaut und auf dieser Grundlage die Bauelementebasis bis 1990 erweitert. Das gilt insbesondere für die Bereitstellung moderner 16- und 32-Bit-Mikroprozessorsysteme sowie höchstintegrierter Speicherschaltkreise bis zu 1 M Bit Speichervermögen und von kundenspezifischen Schaltkreisen unter Einbeziehung der Anwender.

Darauf aufbauend werden vom Kombinat Robotron im gleichen Zeitraum mindestens 170000 Büro- und Personalcomputer, 1950 Kleindatenverarbeitungsanlagen und 670 elektronische Datenverarbeitungsanlagen bereitgestellt. Hinzu kommt ein breites Sortiment moderner peripherer Geräte von Folien- und Festplattenspeichern, alphanumerischen und grafischen farbigen und Schwarz-weiß-Bildschirmgeräten sowie von Druckern und Zeichengeräten, miteinander koppelfähig über Datennetze.

Zur Versorgung der Volkswirtschaft mit hochwertigen elektronischen Erzeugnissen der Automatisierungstechnik werden auf der gleichen Bauelementebasis neue Generationen numerischer und nichtnumerischer Steuerungen für kontinuierliche und Montageprozesse von Robotersteuerungen, Antriebssteuerungen, u. a. von Drehstromantrieben, und Informationsverarbeitungseinrichtungen von den Kombinaten des Industriebereiches Elektrotechnik und Elektronik sowie des Werkzeug- und Verarbeitungsmaschinenbaus bereitgestellt.

In den Betrieben und Kombinaten der Elektrotechnik und Elektronik werden bis 1990 etwa 5800 verallgemeinerungsfähige Beispiele für CAD/CAM-Lösungen geschaffen. Die hohe volkswirtschaftliche Effektivität dieser Vorhaben infolge wesentlicher Verringerung der Entwicklungs- und Überleitungszeiten, Erhöhung der Material- und Energieökonomie und der Gebrauchswerte der Erzeugnisse wird daraus ersichtlich, daß bei einer relativen Arbeitskräftefreisetzung je Einsatzfall zwischen 3 und 20, die Aufwendungen sich in 1,5 bis 2 Jahren bereits amortisieren. Die Anwendungschwerpunkte liegen vor allem beim Entwurf von monolithischen und hybriden Schaltkreisen sowie von Leiterplatten, mechanischen Konstruktionsteilen sowie im Werkzeug- und Formenbau, bei der Projektierung im Starkstrom- und nachrichtentechnischen Anlagenbau, bei der technologischen Produktionsvorbereitung bis zur Produktionssteuerung.

Der Umgang mit dem *Mikroprozessor* als dem zentralen Baustein für solche hocheffektiven rechentechnischen und informationsverarbeitenden Geräte und Anlagen der CAD/CAM-Technik und der Automatisierungstechnik stellt hohe Ansprüche an einen ständig wachsenden Teil unserer Menschen und verändert tiefgreifend deren Arbeitsinhalte sowie Arbeits- und Lebensbedingungen.

Die Erschließung der riesigen Effektivitätsreserven aus der Anwendung dieser Schlüsseltechnologien muß einhergehen mit der rechtzeitigen Einbeziehung aller beteiligten Werktätigen, beginnend bei Maßnahmen der Qualifizierung und der Lenkung ihrer vielfältigen Initiativen auf diese neuen Aufgaben.

Vor allem unsere jungen Menschen an den Polytechnischen Schulen, Hoch- und Fachschulen sowie Universitäten müssen zum frühestmöglichen Zeitpunkt mit diesen neuen Fragen der Anwendung rechentechnischer Mittel vertraut gemacht und ihre schöpferischen Fähigkeiten geweckt werden.

In diesem umfassenden Prozeß der Qualifizierung und des Erfahrungsaustausches kommt der Zeitschrift „Mikroprozessortechnik“ als Mittler von Wissen und Erkenntnissen eine wichtige Aufgabe zu.

Dazu wünsche ich der Redaktion der Zeitschrift und ihren Lesern viel Erfolg.



Meier

Felix Meier
Minister für Elektrotechnik
und Elektronik

CMOS-Gate-Array-System U5200

Schaltungstechnik und Anwendung

Manfred Sorst, Dr. Michael Gieseler,
Dr. Wolf-Joachim Fischer
VEB Forschungszentrum
Mikroelektronik Dresden

Die Gate-Array-Technik als ein Verfahren zur Herstellung anwendungsspezifischer Schaltkreise (ASIC) hat eine weite Verbreitung gefunden. Dabei wird auf Grund der erreichbaren hohen Störsicherheit sowie der niedrigen Gatterverlustleistung bevorzugt die CMOS-Technologie /1/ eingesetzt. Für das Gate-Array-System U5200 kommt eine 4-µm-CMOS-Technologie mit zwei Polysiliziumebenen und einer Metallisierungsebene zur Anwendung. Dabei werden die obere Polysiliziumebene, eine Kontaktfensterebene sowie die Metallisierungsebene für die Overlays der Makros und die kundenspezifische Kanalverdrahtung genutzt. Die Haupteigenschaften des Systems U5200, der Grundaufbau des Masters und der Zellen sowie die Leistungsparameter des zugehörigen CAD-Systems „ARCHIMEDES" wurden an anderer Stelle /2, 3/ bereits ausführlich beschrieben.

1. Konstruktion der Gate-Array-Zelle

Die U5200-Gate-Array-Zelle (Bild 1.1) enthält insgesamt acht Transistoren (4 NMOS, 4 PMOS) und zwei Durchführungen. Das für dieses Gate-Array-System zur Anwendung kommende LSSD-Verfahren zur Berechnung der Testfolgen setzt voraus, daß die Speicherung von Informationen ausschließlich in den dafür vorgesehenen Master-Slave-Flip-Flops erfolgt. Eine Informationsspeicherung durch Ladungsspeicherung auf parasitären Kapazitäten ist daher nicht möglich. Aus diesem Grund wird für das U5200-System eine voll statische CMOS-Schaltungstechnik ohne Transfergate-Transistoren angewendet. Vorteilhafterweise werden in einem solchen Fall die Gates jeweils eines NMOS- und eines PMOS-Transistors der Zelle bereits im Masteruntergrund durch das Polysilizium der Gates fest verbunden (Bild 1.1). Das Wesen der Gate-Array-Technik bedingt eine einheitliche Konstruktion aller Zellen eines Typs. Es ist eine Optimierung der Transistorparameter bezüglich des dynamischen Verhaltens nur für einen Macrotyp möglich. Die elektrischen Parameter für alle anderen Macrotypen liegen damit ebenfalls fest. Die U5200-Zelle wurde für Gatter mit zwei Eingängen (NAND2, NOR2) optimiert. Unter der Voraussetzung, daß die Verzögerungszeiten tvLH, tvHL des NAND2 für einen High-Low- und Low/High-Übergang am Gatterausgang gleich sind, gilt für das Verhältnis der Breiten Wp des PKOS- und Wn des NKOS-Transistors näherungsweise

$$\frac{Wp}{Wn} = \frac{Kn}{Kp} \cdot \frac{(Ucc - UTn)^2}{(Ucc - UTp)^2} \qquad (1)$$

Dabei sind Ucc die Betriebsspannung, UTn und UTp die Schwellspannungen des NMOS- bzw. PMOS-Transistors, Kn und Kp die zugehörigen Transistorkonstanten. Für typische Parameter ergibt sich ein Verhältnis von ungefähr zwei. Die vom Logikentwerfer zu beachtende Folie einer solchen Optimierungsstrategie ist eine Bevorzugung der NAND-Macros gegenüber den NOR-Macros (Bild 1.2).

2. JK-Master-Slave-Flip-Flop mit Testservice und Taktversorgung

Rückführungen innerhalb der Anwenderkombinatorik sind wegen der vollautomatischen Testpatternerzeugung für den Strukturtest der Schaltung unzulässig. Der Anwender muß daher auf vorgefertigte sequentielle Schaltelemente zurückgreifen können. Diese werden ihm in Gestalt von funktionsspezialisierten und dynamisch optimierten JK-Master-Slave-Flip-Flops angeboten, die durch ihre Struktur die automatische Testpatternerzeugung gestatten. Das Masterkonzept basiert auf der intermittierenden Anordnung von je zwei Gatterzeilen und einer Flip-Flop-Zeile. Auf Grund der konzentrierten Zeilenanordnung werden alle nicht anwenderspezifischen Signale fest verdrahtet angeboten.

Diese sind:

- das Taktsignal C
- das Modulsteuersignal M
- das Schiebeeingangssignal QSI sowie das Schiebeausgangssignal QSO.

Taktsystem

Sämtliche Flip-Flops werden mit dem Systemtakt versorgt. Das Logikkonzept des Anwenders muß sich daher auf syn-

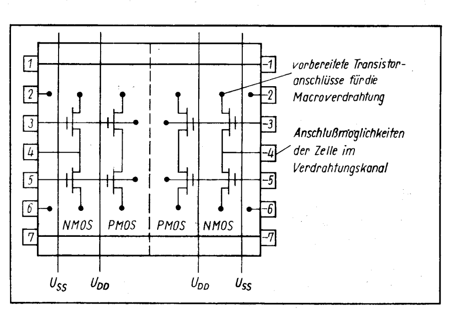


Bild 1.1 U5200-Zelle

Macro	tv LH/ns	tvH1/ns
NAND 2	6,1	5,0
NOR 2	9,3	4,1
NAND 3	6,7	10,3
NOR 3	15,4	4,2
NAND 4	7,3	13,6
NOR 4	21,0	4,4

Bild 1.2 Verzögerungszeiten für NOR- und NAND-Gatter

chrone Schaltungen beschränken. Im Punkt 4. wird gezeigt, daß dies keine Einschränkung der Anwendungsbreite bedeutet.

Der Taktpfad verläuft vom Eingangspegelwandler (siehe 3.) über den Treiber der Taktverteilung zu den Zeilentreibern. Bild 2.1 zeigt die Struktur des Taktsystems.

Ein zentrales Problem stellt der Taktskew dar.

Bild 2.2 vermittelt einen Eindruck über die Taktlaufzeiten auf dem Chip bei etwa 80 Prozent Auslastung und unterschiedlicher Belegung der Matrixhälften.

JK-Master-Slave-Flip-Flop mit Testservice

In Bild 2.3 ist der Logikplan des Flip-Flops dargestellt. Folgende Funktionen sind durch das Modussteuersignal M möglich:

– M = LOW: Anwendermodus:

Die Schaltung arbeitet als JK-Master-Slave-Flip-Flop bezüglich der Eingänge J, K, R, S sowie der Datenausgabe über Q und $\bar{Q}$ des Arbeitsslaves SA. Der Schiebeslave SS vollzieht alle Zustandsänderungen mit und enthält damit stets die aktuelle Ausgangsinformation. Das Signal an QSI wird ignoriert.

– M = HIGH: Schiebemodus:

Die Eingänge J, K, R, S sind unwirksam;

der Slave SA ist vom Master abgetrennt und hält die eingespeicherte Information. Die Stufe arbeitet als RS-Master-Slave-Flip-Flop bzgl. QSIi und QSOi. Alle QSOi sind mit dem folgenden QSIi + 1 in der Zeile fest als Schieberegister verdrahtet. Dies erlaubt ein serielles Beschreiben bzw. Auslesen aller Flip-Flops während der Schaltkreistestung. Neben den oben dargestellten Betriebsarten mit stationären M werden innerhalb des Tests auch Datenwege genutzt, die gezielt Daten in einen der beiden Slaves einbringen können.

– LOW → HIGH-Flanke an M während der Master-Phase des Flip-Flops (Takt C = LOW) führt zur Übernahme eines Arbeitstaktergebnisses in den Schiebeslave SS, wobei der Arbeitsslave unbeeinflußt bleibt. Diese Ergebnisse können über QSO seriell ausgelesen werden.

– HIGH → LOW-Flanke an M während der Slave-Phase (C = HIGH) führt zur Übernahme eines Schiebeergebnisses in den Arbeitsslave SA und ermöglicht eine Initialisierung über QSI.

Die Nutzung dieser Datenkopplung zwischen dem Master und beiden Slaves bleibt ebenfalls dem Schaltkreistest vorbehalten. Neben der Effektivierung des Prüfablaufs dient der Schiebeslave SS dem Erkennen speichernder Leitungen beim Test. Wie Bild 2.3 zeigt, sind Master und Slaves mit invers zu aktivierender Logik (nur ein Takt) aufgebaut. In allen Darstellungen sind die Wannen- und Substratanschlüsse aus Übersichtsgründen unbeschaltet geblieben. Auf dem Chip sind alle p-Wannen auf Masse und das n-Substrat auf Betriebsspannung UCC geklemmt.

Zusammenspiel zwischen Taktverteilung und Flip-Flops

Durch die dynamischen Parameter von Taktverteilung und Flip-Flop und den längsten Pfad der Anwenderkombinatorik wird die maximale Taktfrequenz fTmax eines verdrahteten Gate-Arrays bestimmt. Dabei muß vom folgenden Ablauf ausgegangen werden:

– Während der Slave-Phase laufen alle Ausgleichsprozesse in der Kombinatorik ab. Das bedeutet, daß in dieser Phase auch die Eingangsinformationen bereitgestellt sein müssen.

– 40 ns vor der Öffnung des Masters (HIGH-LOW-Flanke an C) müssen alle dynamischen Vorgänge abgeschlossen sein (SET-UP-Zeit der Flip-Flops).

– In der Master-Phase werden die Daten übernommen.

Für die maximale Taktfrequenz des verdrahteten Schaltkreises fTmax gilt die Gleichung 2.

Bild 2.4. veranschaulicht die zeitlichen Verhältnisse innerhalb einer Taktperiode.

Im Bild sind

– anwenderunabhängige Zeiten (mit * gekennzeichnet):

tSW
Taktskew
tSU
SET-UP-Zeit der Flip-Flops
i/(2 fmax)
minimale Takt-LOW-Phase

$$fTmax = \frac{1}{2\,tSW + tINP + tLK + tSU + (1/(2\,fmax))} \qquad (2)$$

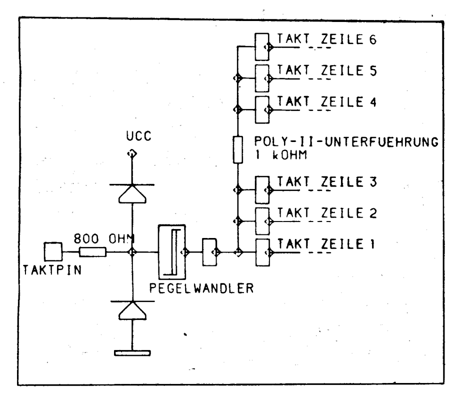


Bild 2.1 Struktur des Taktsystems

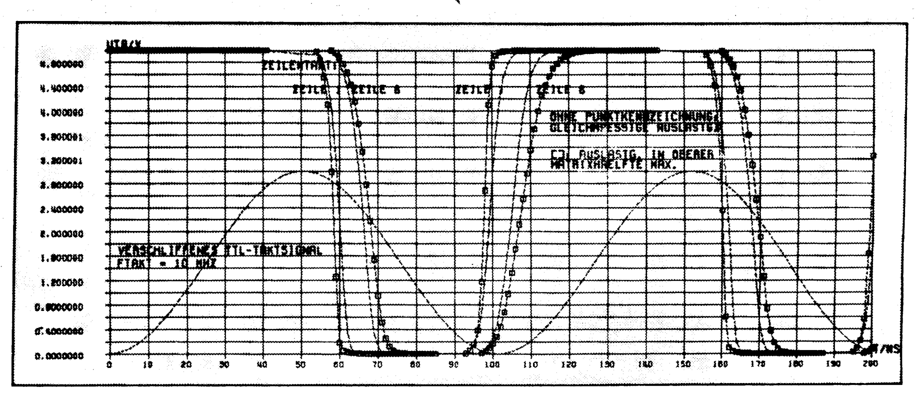
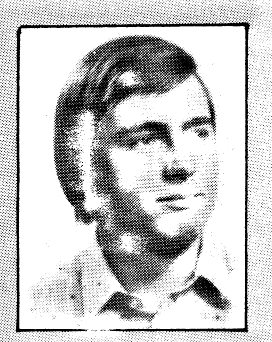
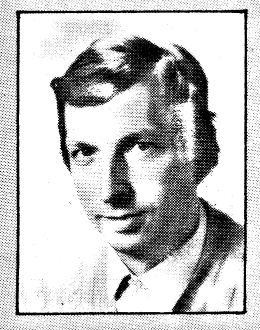


Bild 2.2 Dynamisches Verhalten des Taktsystems U5201

Manfred Sorst wurde 1955 in Leipzig geboren. Er studierte von 1975 bis 1979 an der Sektion Informationstechnik der Technischen Universität Dresden im Bereich Technische Kybernetik und Automatisierungstechnik. In seiner Diplomarbeit beschäftigte er sich mit Mikrorechner-Reglern.
Zur Zeit arbeitet er im VEB ZFTM Dresden an der Entwicklung des Gate-Array-Systems U5000.



Michael Gieseler, Jahrgang 1952, wurde in Dresden geboren. Nach Abschluß des Abiturs und der Facharbeiterausbildung als Funkmechaniker studierte er an der Sektion Informationstechnik der Technischen Universität Dresden und diplomierte 1974. Von 1974 bis 1978 arbeitete er als wissenschaftlicher Assistent im Bereich Bauelemente und Systeme der TU Dresden. Mit einer Arbeit zur automatischen Layoutgenerierung von I^2L-Schaltungen promovierte Michael Gieseler 1978 zum Dr.-Ing. Seit 1979 arbeitet Dr. Michael Gieseler als Entwurfsingenieur im VEB Zentrum für Forschung und Technologie Mikroelektronik Dresden. Zur Zeit arbeitet er auf dem Gebiet des Gate-Array-Entwurfes.



Wolf-Joachim Fischer (36) ist gebürtiger Dresdener. Er studierte Informationstechnik an der TU Dresden und diplomierte 1973 mit einer Arbeit auf dem Gebiet der Modellierung von MOS-Transistoren. Mit Arbeiten zur MOS-Schaltungstechnik sowie zur Analyse des innerelektronischen Verhaltens von MOS-Transistoren erwarb er 1976 bzw. 1983 die akademischen Grade Dr.-Ing. und Dr. sc. techn.
Seit 1973 arbeitet Dr. Fischer im Schaltungsentwurf des VEB Zentrum für Forschung und Technologie Mikroelektronik, gegenwärtig auf dem Gebiet der Entwicklung von Gate-Array-Systemen.



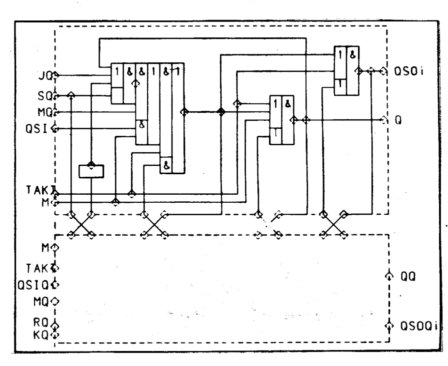


Bild 2.3 Logikplan des Flip-Flops

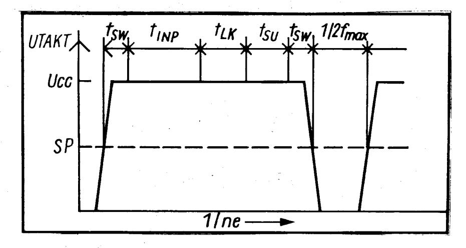


Bild 2.4 Zeitaufteilung in einer Taktperiode

– anwenderspezifische Zeiten:
tINP
zusätzliches Zeitfenster, in der externe Eingaben erfolgen können
tLK
Laufzeit der längsten Kombinatorikkette

3. Das Interfacesystem

Das Interfacesystem ist bei allen anwenderspezifischen Stufen gleichartig aufgebaut und setzt sich aus den Anteilen
– Bondinsel mit Schutzschaltung gegen elektrostatische Aufladung
– Eingangspegelwandler mit Treiber
– Ausgangsstufe mit Testservice
zusammen. Durch die Anwenderverdrahtung wird die funktionelle Leistungsfähigkeit einer Interfacestufe bestimmt. Es können reine Eingänge, reine Ausgänge und bidirektionale Interfaces realisiert werden.

Eingangspegelwandler mit Treiber

Direkt an die Schutzschaltung sind die Gates der Eingangsstufe (T1 ... T4 in Bild 3.1) angeschlossen. Diese Schaltung sichert die eingangsseitige Schaltpunktverschiebung auf 1.5 ... 1,6 V, wodurch sowohl CMOS- als auch TTL-Eingangspegel verarbeitet werden können. Die Schaltpunktverschiebung wird durch eine p-Kanal-transistorseitige Gegenkopplung erzielt, ohne dabei die dynamischen Nachteile unsymmetrisch dimensionierter Inverter oder eines Vergleiches mit einem Referenzpegel in Kauf nehmen zu müssen. Da die Lastunabhängigkeit der Signale auf dem Gate-Array ein wichtiges Problem darstellt, ist dem Pegelwandler noch ein Zweifachinverter als Treiber nachgeschaltet. Dieser ist mit den Transistoren T5 ... T8 realisiert.

Ausgangsstufe mit Testservice

Die Maximalausbaustufe stellt eine tristatesteuerbare Endstufe dar, die neben dem Dateneingang Ea und dem Output-Enable-Eingang $\overline{OE}$ zwei zusätzliche, dem Anwender nicht zugängliche Steuereingänge S0 und S1 besitzt, die unabhängig von der Belegung von Ea und $\overline{OE}$ den Ausgang LOW, HIGH oder hochohmig schalten können. Bild 4.1 gibt die logischen Möglichkeiten an. Das Bild 3.2 zeigt die schaltungstechnische Realisierung dieser Endstufe. Der genannte Testservice trägt wesentlich zur Effektivierung der Messung bei, wozu alle S0- und S1-Eingänge zentral versorgt werden. Somit ist durch die Gleichschaltung aller Ausgänge eine

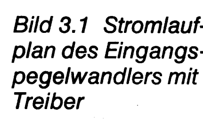


Bild 3.1 Stromlaufplan des Eingangspegelwandlers mit Treiber

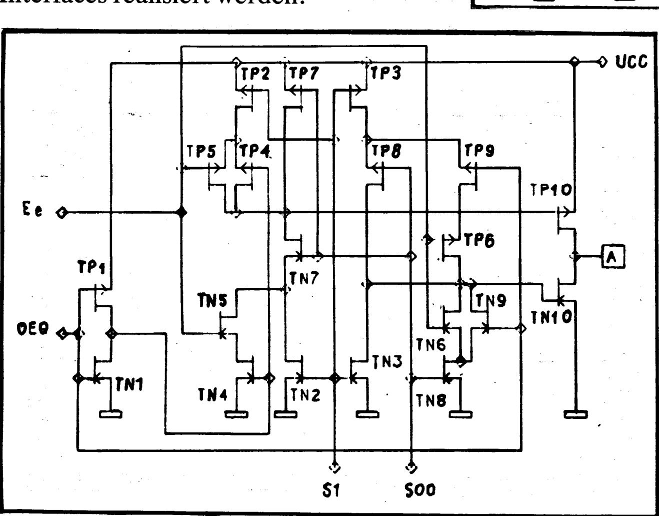


Bild 3.2 Stromlaufplan der Ausgangsstufe

S0	S1	$\overline{OE}$	Ea	Ausgangssignal
L	L	L	L	DATEN von Ae
		L	H	DATEN von Ae
		H	*	KUNDEN-TRI-STATE
L	H	*	*	PRUEF-HIGH
H	L	*	*	PRUEF-LOW
H	H	*	*	PRUEF-TRI-STATE

Bild 4.1 Codierung der Steuersignale S0 und S1

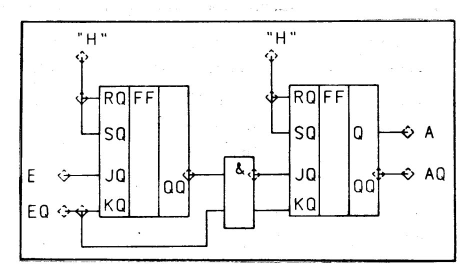


Bild 4.2 Schaltung FLAEIN zur Ereignisauswertung

schnelle Erfassung der statischen Parameter möglich.
Weiterhin erlauben die an speziellen Pins zugänglichen Steuersignale S0 und S1 dem Anwender einen In-Circuit-Test von Leiterplatten, die U5200-Schaltkreise enthalten.

4. Anwendung

Die Realisierung der anwenderspezifischen Logik erfolgt auf der Basis der vom Hersteller entworfenen Hardwaremacros (Logikgatter, Flip-Flop, Interfacestufen).
Darüber hinaus werden Beispiellösungen komplexerer Schaltungen bereitgestellt.
Im synchronen System steht dem Logikentwerfer kein Takteingang an den Flip-Flops zur Verfügung. Alle Master bzw. Slaves der Flip-Flops übernehmen die an ihren Eingängen anliegende Information nur während einer der Taktphasen. Somit reduziert sich das Problem des Logikentwurfes sequentieller synchroner Schaltungen auf die Berechnung der Ansteuerfunktionen für die Logikeingänge der Flip-Flops unter Einbeziehen ihres logischen Zustandes im vorangegangenen Taktzyklus.
Aus der genannten Tatsache ergibt sich, daß alle Zeitabläufe in einer festen Beziehung zum am Gate-Array anliegenden Takt C stehen, sobald in ihrem Signalweg ein Flip-Flop liegt. Derartige Signale sind taktsynchron. Andererseits ergibt sich für eine definierte Arbeitsweise eines solchen Gate-Arrays, daß alle von außen an das Gate-Array führenden Eingangssignale, welche auf getaktete Strukturen führen (Zähler, Teiler, Register), den Synchronbedingungen genügen müssen oder aber auf dem Chip zu synchronisieren sind.

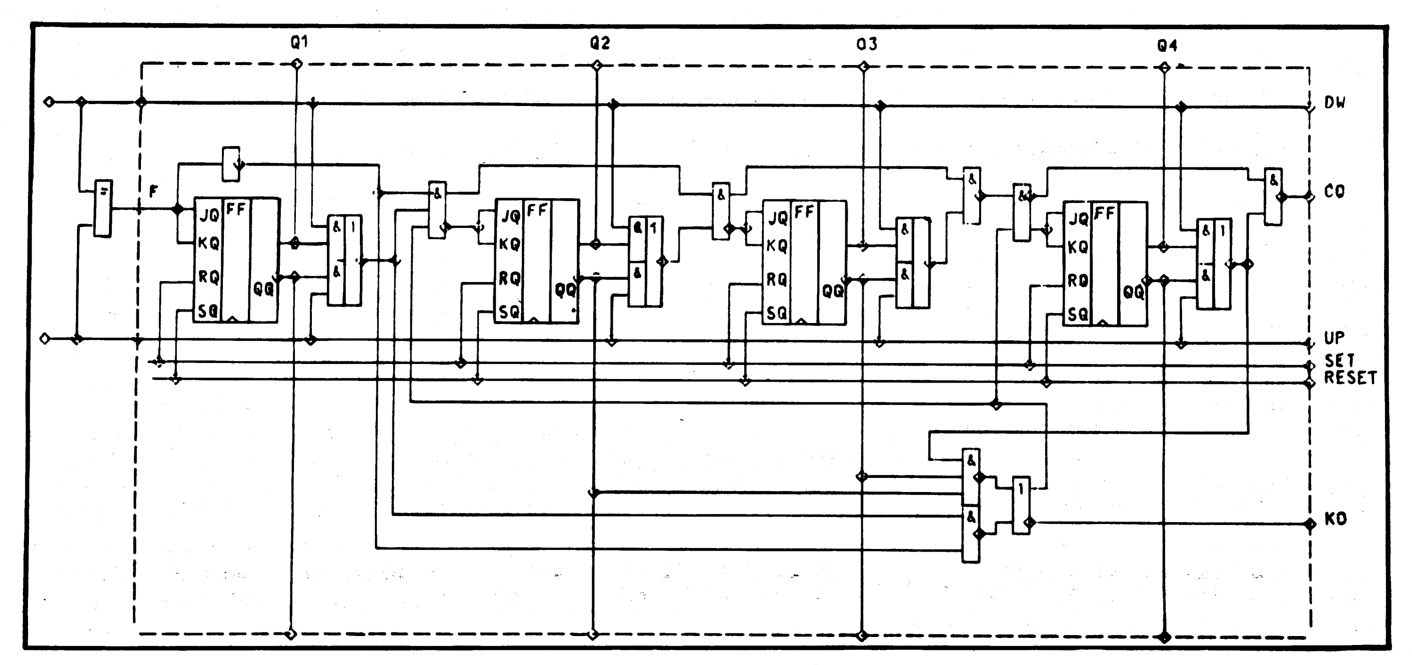


Bild 4.3 Dekadischer Zähler bezüglich Gate-Array-Takt C

Eine logisch sichere Methode für die Ansteuerung einer internen sequentiellen Schaltung mit nicht synchronen Eingangssignalen ist die Flankenauswertung (Bild 4.2). Im gezeigten Beispiel wird ein beliebig langer, aber wenigstens eine Takt-LOW-Phase überdeckender HIGH-Impuls (unter zusätzlicher Beachtung der SET-UP-Zeit der Flip-Flops) am Eingang „EIN" auf genau eine, und zwar die 2. der dem Ende des HIGH-Impulses folgenden HIGH-LOW-Flanke folgende Takt-Periode, beginnend mit TAKT C = HIGH, am Ausgang AUS verkürzt. Zwischen den HIGH-Ereignissen an EIN muß jeweils mindestens eine, die nächste volle Taktperiode (beginnend mit TAKT C = „HIGH") überdeckende, LOW-Phase folgen.
In Bild 4.3 ist ein umschaltbarer dekadischer Vor- und Rückwärtszähler mit Setz- und Rücksetzmöglichkeit dargestellt. Die Zählung bezieht sich auf den Eingang CI, an dem über das Gatter G14 (EXAND) der Zähleingang J = K mit LOW freigegeben wird. Es werden demzufolge die am Takteingang C des GATE-ARRAY anliegenden Taktimpulse während CI = LOW direkt gezählt.
Die zwischen den für die 10 logischen Zustände notwendigen 4 Flip-Flops befindliche Logik stellt die Ansteuerfunktionen für die als T-Flip-Flop verwendeten JK-Master-Slave-Flip-Flops entsprechend der Zählaufgabe (vor/rück/Stop) bereit und realisiert die „0"- bzw. „9"-Erkennung für die entsprechende Übertragsweiterleitung.
Im Unterschied zur im Bild 4.3 gezeigten Art der Ansteuerung des Zähleinganges ist ebenso die Ansteuerung mit der in Bild 4.2 angegebenen Schaltung möglich, wobei der negierende Ausgang der Flankenauswerteschaltung an CI

S: COUD1 1. 7.1985 09:11119 M: MODE 0 Takt:	RS EE ST	DU WP	CK II	QQQQ 1234	NNNN QQQQ 1234	C O	K O
2:	HH	LH	LH	LLLL	HHHH	H	L
3:	LH	LH	LH	HLLL	LHHH	H	L
4:	LH	LH	LH	LLLL	HHHH	H	L
5:	LH	LH	LH	LLLL	HHHH	H	L
6:	HH	LH	LH	LLLL	HHHH	H	L
7:	HH	LH	LH	HLLL	LHHH	H	L
8:	HH	LH	LH	LHLL	HLHH	H	L
9:	HH	LH	LH	HHLL	LLHH	H	L
10:	HH	LL	HL	LLHL	HHLH	H	L
11:	HH	LL	HL	LLHL	HHLH	H	L
12:	HH	LH	LH	LLHL	HHLH	H	L
13:	HH	LH	LH	HLHL	LHLH	H	L
14:	HH	HH	HL	LHHL	HLLH	H	L
15:	HH	HH	HL	LHHL	HLLH	H	L
16:	HH	LH	LH	LHHL	HLLH	H	L
17:	HH	LH	LH	HHHL	LLLH	H	L
18:	HH	LH	LH	LLLH	HHHL	H	L
19:	HH	LH	LH	HLLH	LHHL	L	H
20:	HH	LH	LH	LLLL	HHHH	H	L
21:	HH	LH	LH	HLLL	LHHH	H	L
22:	HH	LH	LH	LHLL	HLHH	H	L
23:	HH	LH	LH	HHLL	LLHH	H	L
24:	HH	LH	LH	LLHL	HHLH	H	L
25:	HH	LH	LH	HLHL	LHLH	H	L
26:	LH	LH	LH	LHHL	HLLH	H	L
27:	LH	LH	LH	LLLL	HHHH	H	L
28:	LH	LH	LH	LLLL	HHHH	H	L
29:	LH	LH	LH	LLLL	HHHH	H	L
30:	HH	LH	LH	LLLL	HHHH	H	L
31:	HH	HL	LH	HLLL	LHHH	H	L
32:	HH	HL	LH	LLLL	HHHH	L	H
33:	HH	HL	LH	HLLH	LHHL	H	L
34:	HH	HL	LH	LLLH	HHHL	H	L
35:	HH	HL	LH	HHHL	LLLH	H	L
36:	HH	HL	LH	LHHL	HLLH	H	L
37:	HH	HL	LH	HLHL	LHLH	H	L
38:	HH	HL	LH	LLHL	HHLH	H	L
39:	HH	HL	LH	HHLL	LLHH	H	L
40:	HH	HL	LH	LHLL	HLHH	H	L
41:	HH	HL	LH	HLLL	LHHH	H	L
42:	HH	HL	LH	LLLL	HHHH	L	H
43:	HH	HL	LH	HLLH	LHHL	H	L
44:	HH	HL	LH	LLLH	HHHL	H	L

Bild 4.5 Simulationsprotokoll für den dekadischen Zähler nach dem Abtastprinzip

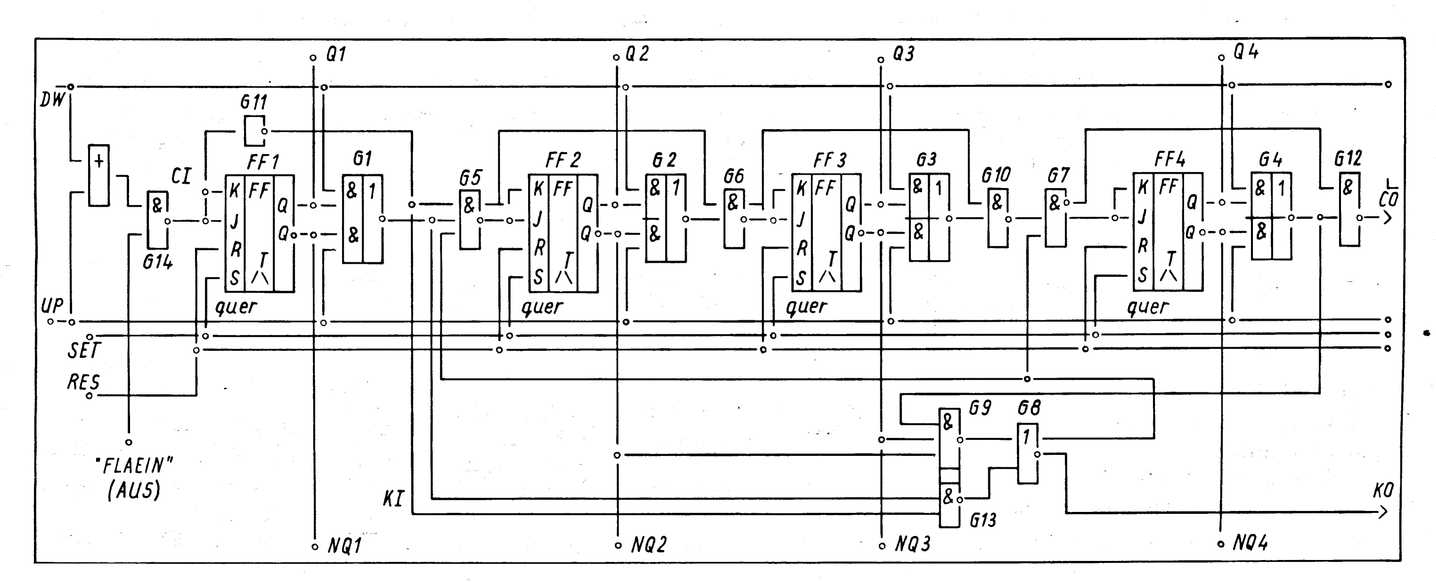


Bild 4.4 Dekadischer Zähler nach dem Abtastprinzip

des Zählers angeschlossen werden muß, da CI selbst Eingang der LOW-aktiven Flip-Flops ist; die Negation wird dabei im folgenden NAND2-Tor realisiert, in das auch die Stop-Funktion für das 1. Bit eingreift (Bild 4.4). Das Zählergebnis ist dann ein „Abtastergebnis" der Eingangsereignisse an einem beliebigen Logikeingang des GATE-ARRAY entsprechend dem Verhalten der Flankenauswerteschaltung. Die Genauigkeit wird somit vom Verhältnis zwischen Eingangsfrequenz und Taktfrequenz C bestimmt. Im Simulationsprotokoll Bild 4.5 des GATE-ARRAY-SYSTEM-eigenen Simulators ist ein Zählbeispiel der beschriebenen Schaltung mit Flankenauswertung nach Bild 4.4 dargestellt. Es besteht somit die Möglichkeit, auf einem Chip verschiedene Frequenzen nach der Abtastmethode zu verarbeiten, wobei durch den gemeinsamen Abtasttakt C alle Ausgangssignale in einem entsprechenden (synchronen) Raster erscheinen. Die höchste verarbeitbare Frequenz wird dabei durch den maximal möglichen Takt selbst bestimmt.

Literatur:

/1/ Zimmer, G.: MOS-Technologie. R. Oldenbourg-Verlag München, Wien 1982.

/2/ Fischer, W. J.; Gieseler, K.; Arndt, W.; Sorst, K; Köhler, T.: CMOS-Gate-Array-System U5200. Nachrichtentechnik, Elektronik 36 (1986) 1

/3/ Fischer, W. J.; Knobloch, J.: Maßanzug für Schaltkreise – Gate-Arrays und Standard-Zellen-ICs für den Mittelstand. Konstruktion, Elemente, Methoden KEM (1986) 6. Leinfelden-Echterdingen

/4/ Friedman, E.; Powell, S.: Design and Analysis of a Hierarchical Clock Distribution System for Synchronous Standard Cell/Macrocell VLSI. IEEE J. Solid-State Circuits CS-21 (1986) 2. New York.

Das Echtzeitbetriebssystem IRTS 8000

Peter Bala, Rainer Haupt,
Dr. Ludwig Claßen
Zentrum für Forschung und Technologie, Kombinat VEB Elektro-Apparate-Werke Berlin-Treptow

Echtzeitbetriebssysteme sind dadurch gekennzeichnet, daß der Rechner, auf dem sie abgearbeitet werden, auf echte, real existierende zeitkritische Ereignisabläufe in seiner Umwelt reagieren muß. Als ein entscheidendes Maß für die Leistungsfähigkeit eines Rechners mit einem Echtzeitbetriebssystem ist deshalb die Reaktionszeit anzusehen (im Mikrosekundenbereich), die beim Umschalten (task switch) von einem Prozeß (Programm) mit niedriger Priorität auf einen Prozeß (Programm) mit höherer Priorität (z. B. Uhrimpuls, Beendigung einer Ein-/Ausgabeaktivität, Alarmmeldung) vergeht.
Das hier beschriebene Echtzeitbetriebssystem IRTS 8000 ist ein durch den Anwender konfigurierbares Standard-Softwaresystem für Echtzeit-Mikrorechner. Es unterstützt die 16-Bit-Mikroprozessorfamilie U8001/U8002 und besteht aus einem relativ kleinen Systemkern von 4 KByte, um den anwendungsspezifische Ergänzungsmodule wie eine formatgesteuerte Ausgabeorganisation, Terminal-/Druckerhandler, Testhilfsmittel u. a. m. angeordnet werden können.

1. IRTS 8000-Übersicht

IRTS (INTEGRABLE REAL TIME SOFTWARE) ist ein vom Anwender konfigurierbares Softwaresystem mit Multitaskeigenschaften für Echtzeitanwendungen. Es besteht aus einem relativ kleinen Echtzeitsystemkern (Kernel), um den system- und anwendungsspezifische Ergänzungsmodule angeordnet werden. Der Systemkern organisiert die Steuerung und Synchronisation von multiblen Systemanforderungen in einer Echtzeitumgebung.
Er enthält die Echtzeitbasisfunktionen
- Tasksteuerung
- Semaphoresteuerung
- Intertaskkommunikation (Mailboxes)
- Echtzeituhr
- Memory Management.

IRTS kann sowohl in einem RAM- als auch in einem EPROM-basierenden Anwendersystem eingesetzt werden. Es unterstützt die Mikroprozessoren der Familie U8000 (U8001 und U8002).
Als typische Umschaltzeit vom aktuell bearbeiteten Prozeß auf einen Prozeß mit anderer Priorität (Task-Wechsel) hat sich bei IRTS ein Wert von etwa 250 Mikrosekunden erwiesen. Beim Eintreffen von externen Ereignissen (Interrupts) findet kein Taskwechsel statt, so daß die Interruptserviceroutinen als Teil der aktuell laufenden Task abgearbeitet werden, sofern nicht Systemrufe einen Prioritätswechsel erzwingen.
Die Systemkonfigurierung von IRTS erfolgt rechnergestützt mit dem Sprachprozessor ICL (IRTS Configuration Language).

Zur Installation von IRTS wird neben dem Mikroprozessorsystem (CPU, RAM- bzw. EPROM-Speicher) ein interruptfähiger Zeitgeberbaustein zur Realisierung der Echtzeituhr und bei Anwendung des IRTS-Debuggers ein interruptfähiger Zählerbaustein benötigt, mit dem der CPU-Status Stack-Memory Request beim Single-Step-Betrieb gezählt werden kann. Zur Unterbringung des Systemkerns von IRTS ist ein Speicherbereich von etwa 4 KByte vorzusehen.
Als spezielle Beispielapplikationslösung existiert eine Implementation des Echtzeitbetriebssystems IRTS auf dem Programmier- und Entwicklungssystem P8000[1]).
Das Konzept des Echtzeitsteuerprogrammsystems IRTS basiert auf den drei Basiselementen Tasks, Semaphore sowie Mailboxes und Messages.
Tasks werden verwendet, um die Arbeit des Prozessors in möglichst unabhängige Teilkomponenten aufteilen zu können und damit eine optimale Zeitbilanz auf einem Prozessor zu erzielen. Eine Task besteht aus einem Programm zusammen mit einem bestimmten CPU-Zustand (Registerinhalte, Flag-Belegungen

[1]) Das Programmier- und Entwicklungssystem P8000 aus dem Kombinat VEB Elektro-Apparate-Werke Berlin-Treptow stellen wir im Heft 3/1987 ausführlich vor.

Peter Bala (46) studierte von 1960–1966 in der Fachrichtung Regelungstechnik an der TH Ilmenau. Seit 1966 Mitarbeiter im Rechenzentrum des damaligen Instituts für Regelungstechnik, dem heutigen ZFT des KEAW. Seit 1978 in der Abteilung Basissoftware schwerpunktmäßig mit Echtzeitsoftwaresystemen beschäftigt: ab 1982 Entwicklung des IRTS 8000.

Reiner Haupt (43) studierte von 1970 bis 1975 Maschinenbau an der TH Karl-Marx-Stadt. Seit 1970 im Institut für Regelungstechnik bzw. ZFT des KEAW tätig, befaßt er sich seit 1979 in der Abteilung Basissoftware mit der Entwicklung von Echtzeitsoftwaresystemen; ab 1982 mit der Entwicklung des IRTS 8000.

Dr. Ludwig Claßen (43) studierte von 1964–1970 in der Fachrichtung Regelungstechnik an der TH Karl-Marx-Stadt und promovierte 1971. Seit 1971 im Institut für Regelungstechnik, heute ZFT des KEAW, tätig. Von 1973 an Aufbau eines Fachgebietes Mikroprozessorsoftware und seit 1980 Leiter der Abteilung Basissoftware.

usw.). Die Aufteilung in möglichst unabhängige Operationen, denen einzelne Tasks zugeordnet werden, hat den Vorteil, daß das aus dem Zusammenwirken von Teilkomponenten bestehende Gesamtsystem leichter verstanden und behandelt werden kann – vor allem auch in bezug auf den zeitlich (CPU-intern) rein sequentiellen, nach außen jedoch scheinbar parallelen Programmablauf. Wichtig für die Arbeitsweise von IRTS ist, daß alle Tasks nach dem Systemstart laufbereit sind. Die Tasks müssen sich in der Folgezeit selbsttätig mit Hilfe der Dienstleistungen des IRTS-Kerns (System-Calls) verwalten.

Semaphore werden zur Sperr-Synchronisation der Aktionen verschiedener Tasks (z. B. konkurrierende Tasks mit gemeinsamen Variablen) verwendet. Die Synchronisation mit Hilfe von Semaphores besteht dabei darin, eine zeitlich definierte Einordnung dieser Aktionen zu erreichen. Ein Semaphor ist eine Datenstruktur, die eine Aussage darüber erlaubt, welche Tasks auf die Ausführung einer bestimmten Operation warten. Eine ganz typische Anwendung finden Semaphore bei der Verhütung von Zugriffskonflikten unterschiedlicher Tasks auf eine gemeinsame Ressource (RAM-Bereiche mit gemeinsamen Daten, Drucker usw.).

Mailboxes sind ein Werkzeug zur Abwicklung der Intertaskkommunikation. Angewendet werden Mailboxes, wenn eine Information von einer Tasks an eine andere Task zu übermitteln ist oder wenn eine Ereignissynchronisation zweier Tasks mit einem Austausch einer beliebigen Menge von Informationen erfolgen soll. Die Information selbst wird in einem besonderen Nachrichtenträger, dem sog. Messageobjekt (kurz: Message), transportiert. Eine Mailbox ist dabei der Ort, wo Messages für eine andere Task hinterlegt und abgeholt werden können. In IRTS existieren spezielle Operationen, die das Senden einer Message an eine Mailbox und das Empfangen einer Message aus einer Mailbox in vielfältiger Weise unterstützen.

2. IRTS-8000-Systemkern

Der IRTS-Kern ist der Basisblock jedes IRTS-Echtzeitsteuerprogrammsystems. Er enthält die folgenden Echtzeitfunktionen:

- Task Management
- Semaphore
- Clock Management
- Memory Management
- Intertask Communication.

Alle Anforderungen an den IRTS-Kern erfolgen über einen System-Call. Die Parameterübergabe vom Anwenderprogramm an den IRTS-Kern erfolgt in Registern.

Task Management ist der Mechanismus zum Steuern und Überwachen des Ablaufes miteinander konkurrierender Operationen auf einem einzelnen Prozessor. Eine Task besteht aus dem Maschinenkode (Programm), dem Systemmode-Stack, wahlweise einem Normalmode-Stack, dem Taskobjekt (auch als Taskdatenstruktur bezeichnet – einem jeder Task zugeordneten IRTS-Informationsblock zur Taskzustandserfassung und zur Tasksteuerung), dem Stackpointer, dem Pointer auf den nächsten auszuführenden Maschinenbefehl (Programm Counter), dem CPU-Registerzustand und anderen Informationen:

Create Task
Erzeugung einer Task
Destroy Task
Redifinition einer Task
Reschedule Task
Änderung der Taskpriorität
Lock Task
Freigabe des Taskwechsels
Unlock Task
Blockierung des Taskwechsels
Suspend Task
Suspendierung einer Task
Resume Task
Wiederaktivierung einer Task
Wait Task
Selbstsuspendierung einer Task
Who I am
Bereitstellen der Taskobjektadresse
Census of the Task
Ermittlung von Taskinformationen.

Semaphore dienen zur Synchronisation sich gegenseitig beeinflussender Tasks. Ein Semaphor besteht aus einem Zähler, der die unbedienten Anforderungen (Signale) an eine Ressource aufsummiert und aus einer zugehörigen Warteschlange von Tasks, die auf diese Ressource zugreifen wollen.

Ein positiver Zahlwert des Zählers eines Semaphors zeigt an, daß die Ressource frei, d. h. verfügbar ist – der Zahlwert selbst, wie viele Anforderungen sofort gleichzeitig bedient werden könnten. Der Wert Null oder ein negativer Zahlwert des Zählers zeigen an, daß die Ressource nicht verfügbar ist und eine Anforderung in die Warteschlange eingereiht wird. Der negative Zahlwert selbst gibt Auskunft, wie viele Anforderungen (Signale) bereits in die Warteschlange eingereiht wurden. Jede Anforderung dekrementiert den Zähler eines Semaphors – jede bediente Anforderung inkrementiert ihn wieder:

Create Semaphore
Definition eines Semaphors
Destroy Semaphore
Redifinition eines Semaphors
Wait Semaphore
Warten auf ein Semaphorensignal
Signal Semaphore
Setzen eines Semaphorensignals
Test Semaphore
Semaphorenzustandsermittlung
Clear Semaphore
Auflösen eines Semaphors.

Clockmanagement

IRTS arbeitet mit einer interruptgesteuerten Echtzeituhr. Die Zykluszeit dieser Uhr ist abhängig von der gerätetechnischen Realisierung der Interruptquelle. Die interne Software-Uhr wird verwendet für zeitabhängiges Warten von Tasks (timed waits), für zeitabhängiges Aussetzen von Tasks (timeout) und für die zyklisch wechselnde Abarbeitung von Tasks (round robin scheduling). Alle Zeiteinheiten in IRTS werden in ticks (Uhrimpulsen) gemessen und behandelt:

Set Clock
Setzen der Echtzeituhr
Read Clock
Lesen der Echtzeituhr
Clk_Delay_Absolute
Clock-Warteschlange absolut
Clk_Delay_Interval
Clock-Warteschlange relativ.

Memory Management

Das Echtzeitsteuerprogrammsystem IRTS beinhaltet Funktionen zur Verwaltung eines gemeinsamen RAM-Speicherpools für Programme und Datenstrukturen. Diese IRTS-Memory-Management-Funktionen ermöglichen die dynamische Belegung und Freigabe von Speicherbereichen durch die Anwendertasks:

Allocate Memory
Belegung eines Speicherbereiches
Release Memory
Freigabe eines Speicherbereiches
Memory Census
Ermittlung des IRTS-Speicherstatus.

Intertask Kommunikation

Die Intertaskkommunikationsfunktionen von IRTS ermöglichen den Austausch von Informationspaketen (Messages) zwischen unterschiedlichen Tasks. Der Kommunikationsprozeß zwischen zwei Tasks wird in das Senden einer Message durch die eine Task und in das Empfangen einer Message durch die andere Task aufgeteilt. Der Ort, an dem die Message von einer Task hinterlegt wird und eine andere Task auf sie zugreifen kann, wird als Mailbox (Briefkasten) bezeichnet:

Create Mailbox
Definition einer Mailbox
Destroy Mailbox
Freigabe einer Mailbox
Create Message
Definition einer Message
Destroy Message
Freigabe einer Message

Acquire Message
Zuweisung einer Message
Assign Message
Zuweisung initialisierte Message
Send Message
Senden einer Message
Reply Message
Rücksenden einer Message
Receive Message
Empfangen einer Message
Release Message
Freigabe einer Message
Get Message
Ermittlung von Statusinformationen
Read Message
Lesen einer Message
Write Message
Verändern einer Message.

3. Konfigurationssprache ICL

Aufgrund des streng modularen Aufbaus von IRTS ist eine sehr dynamische Anpassung an ganz verschiedenartige Anwendungskonfigurationen möglich. Die Forderung nach einer automatisierten Systemgenerierung wird mit der Verfügbarkeit eines Konfigurations-Sprachprozessors (ICL) erfüllt. Er erlaubt die Verarbeitung von High-Level-Generierungssprachelementen für die notwendigen Hardware-Informationen, für Software-Parameter, für die Linkage Informationen und für die Systemdatenstruktur.
Hauptelemente der ICL-Sprache sind:

CONSTANTS
Spezifizierung von Systemkonstanten
EXCHANGES
Definition einer Applikationsmailbox
FILES
Konfigurations-Link-Dateien
HARDWARE
Beschreibung der Applikationshardware
INITIALIZATION
Spezifizierung der Initialisierungsroutine
INTERRUPT
Definition der Interrupt-Bedingungen
MEMORY
Definition der Speicherkonfiguration
SECTIONS
Spezielle Sektoraufteilungsinformationen
SEMAPHORES
Definition der Applikationssemaphoren
SWITCHES
Flags für den Systemgenerierungsprozeß
TASKS
Definition der Applikationstasks.

4. IRTS-8000-Ausgabeorganisation PRINTF

Die Verwendung einer IRTS-Ausgabeorganisation ist insbesondere für Applikationen auf Assemblerniveau immer dann zu empfehlen, wenn die Kommunikationssoftware übersichtlich, änderungsfreundlich oder projektspezifisch generierbar sein soll.
Das Grundprinzip der Arbeitsweise der Ausgabeorganisation besteht darin, daß eine angegebene „Format"-Anweisung interpretativ abgearbeitet wird und die entstehenden Zeichenketten an das angegebene Gerät weitergereicht werden. Eine Formatanweisung besteht im Normalfall aus einer Zeichenkette mit direkt auszugebendem Text und darin eingelagerten Steueranweisungen.

5. IRTS-8000-Terminal- und Drucker-Handler

Der IRTS-Terminal-Handler besteht aus den zwei weitgehend unabhängigen Teilkomponenten Eingabe-Handler (CONIN-Handler) und Ausgabe-Handler (CONOUT-Handler).
Der CONIN-Handler realisiert die Eingabe von Informationen über ein serielles Interface. Der Handler gestattet neben der normalen Eingabe von Zeichen die Eingabe von Cursor-Steuerfunktionen sowie ESCAPE- und Control-Zeichen. Auf jedem Terminal können dabei quasiparallele Ausgaben in den folgenden drei Arten vorgenommen werden, wobei wahlweise für jede Ausgabeart ein besonderer Bildausschnitt (Window) definiert werden kann:

① Ausgabe des aktuellen Puffers der Tastatureingabe, wobei der Cursor, unabhängig von weiteren Ausgaben auf dem Terminal, bis zum Eingabeende auf der nächsten Eingabeposition verharrt.

② Ausgabe von Vorrangmeldungen.

③ Ausgabe von Informationen, wobei die laufende Ausgabe jederzeit durch die Eingabe von XOFF (Control-S) unterbrochen und mit XON (Control-Q) fortgesetzt werden kann.

Der Drucker-Handler realisiert alle Druckerausgaben unter IRTS. Der Handler selbst wird als eine oder mehrere Taks verwaltet, wobei das Prioritätsniveau der Task(s) vom Anwender festgelegt wird. Die Ausgabe aus einem Anwenderprogramm auf einen Drucker erfolgt über den IRTS-System-Call M_SEND, mit dem ein Nachrichtenträger (Message), der die auszugebende Information transportiert und mit einem speziellen Nachrichtenkode (Request) versehen ist, an die dem Gerät vom Anwender zugeordnete Mailbox versendet wird.

6. IRTS-8000-Testhilfsmittel DEBUGGER/MONITOR

Der IRTS-Debugger ist ein Testhilfsmittel für Anwenderprogramme, die unter der Steuerung von IRTS laufen sollen und deren Echtzeitumgebung während des Testbetriebs weitgehend erhalten bleiben muß. Das bedeutet, daß während des Testbetriebes sowohl die Echtzeituhr als auch bereits betriebsfähige Anwenderprogramme parallel weiterlaufen.
Der Debugger meldet sich nach RESET im Standalone-Modus. In diesem Modus können Programme vom Host-System nachgeladen werden und anwendereigene Initialisierungsroutinen ausgetestet werden. Die Nutzung des IRTS-Debugger ist sowohl im Standalone-Modus als auch im Echtzeitbetrieb möglich.

Dabei existieren folgende Testkommandos:

BREAK-Kommando
Softwarehaltepunkt – das zu testende Programm muß in einem RAM-Speicherbereich geladen sein. Der Debugger verwaltet im Standalone-Modus einen, sonst eine beliebige Anzahl von Haltepunkten. Mit dem Erreichen eines Haltepunktes werden etwaige Zeitaktivitäten eines Programms eingefroren.

GO-Kommando
Das GO-Kommando bewirkt die Fortsetzung des unterbrochenen Programms mit dem aktuellen Stand des Programmzählers.
Wurden in der Zeitspanne, in der sich das zuletzt unterbrochene Programm unter der Steuerung des Debuggers befand, ein oder mehrere Haltepunkte angelaufen, so wird die älteste Unterbrechungsmeldung auf der Konsole angezeigt.

REGISTER-Kommando
Das REGISTER-Kommando dient zur Anzeige und Veränderung von Speicherinhalten, die der unterbrochenen Task zugeordnet sind. Der Inhalt des FCW wird dabei in Mnemoniks angegeben.

DISPLAY-Kommando
Das DISPLAY-Kommando dient zur Anzeige und Veränderung von Speicherinhalten. Die Anzeige kann wahlweise im Byte-, Word- oder Longformat erfolgen.

TRACE/NEXT-Kommando
Unabhängig von weiteren Aktivitäten des gesamten Programmsystems kann ein einzelnes Anwenderprogramm nach dem Anlaufen eines Haltepunktes im Einzelschrittbetrieb durchfahren werden. Im TRACE werden alle Zwischenschritte – ausgenommen die Aktivitäten des Kerns – protokolliert. Am Ende des Kommandos werden die aktuellen Registerinhalte angezeigt.
Im IRTS-Debugger sind außerdem Kommandos zum Lesen/Schreiben von Ports sowie zum Füllen, Vergleichen

und Verschieben sowie zum Laden vom/ zum Hostsystem von Speicher-Inhalten implementiert.
Der IRTS-Monitor ist ein Ergänzungsmodul des IRTS-Debuggers. Er dient in erster Linie der Anzeige von aktuellen Zuständen eines unter IRTS laufenden Anwendersystems. Dabei existieren folgende *Monitorkommandos:*

System-Visite
Je nach Modifikation des Kommandos können der Zustand des Anwendersystems, der Zustand der Speicher-Verwaltung, der Inhalt des Program Status Area und die Systemlaufzeit zur Anzeige gebracht werden.

Task-Historie
Es werden die letzten 16 (beim U8002 : 32) Tasks in der Reihenfolge ihrer Bearbeitung aufgelistet.

Task-Visite
Je nach Modifikation des Kommandos werden alle generierten Tasks, alle laufbereiten Tasks, alle zeitverwalteten Tasks, alle inaktiven Tasks, alle relevanten Informationen einer Task oder die Registerinhalte einer Tasks zum Zeitpunkt der letzten Unterbrechung angezeigt.

Task-Handling
Mit dem Taskhandling können Tasks generiert und zerstört, gestoppt und fortgesetzt werden.

Mailbox-Visite
Je nach Modifikation können alle statisch generierten Mailboxes aufgelistet oder die Auslistung relevanter Parameter der Mailbox sowie die Informationsinhalte der in einer angewählten Mailbox enthaltenen Messages angezeigt werden.

Mailbox-Handling
Mit dem Kommando zum Mailboxhandling können Mailboxes generiert oder zerstört werden; ferner kann an jeweils eine Mailbox eine Nachricht versandt werden.
Semaphor-Visite
Je nach Modifikation können alle statisch generierten Semaphore angezeigt oder der Inhalt der Warteschlange eines Semaphors ausgelistet werden.
Semaphor-Handling
Mit dem Kommando zum Semaphor-Handling können Semaphore generiert, zerstört ode deren Initialwert neu festgesetzt werden.

⊠ KONTAKT ☏

Kombinat VEB Elektro-Apparate-Werke Berlin-Treptow, Zentrum für Forschung und Technologie, Abt. Basissoftware, Hoffmannstr. 15–26, Berlin, 1193, Tel. 4 38 85 32, Koll. Haupt.

Intelligenter Prozeßkoppelmodul

Prof. Dr. Manfred Seifart
Technische Hochschule Magdeburg

Eine der wesentlichen Auswirkungen des Mikroprozessor- und Einchipmikrorechnereinsatzes in der Meß- und Automatisierungstechnik besteht im allmählichen Übergang zu dezentralen prozeßnahen Systemkonzeptionen, wodurch sich eine völlig neue Generation von Meß- und Automatisierungskonzeptionen herausbildet. Hauptbestandteile dieser dezentralen Systeme sind intelligente Module zur Meßgrößenerfassung und zur Beeinflussung des Prozesses.
Der Beitrag beschreibt einen solch universell einsetzbaren programmierbaren Prozeßkoppelmodul (PPM), der mit 7 analogen/binären Eingängen, einem Zähleingang und 7 Ausgängen ausgestattet ist. Charakteristisch für diesen PPM sind der zusätzlich zum klassischen Analogausgang vorhandene galvanisch getrennte Feldbusanschluß (serielle digitale Schnittstelle), seine umfangreiche Programmierbarkeit und Datenvorverarbeitung sowie die äußerst flexible Anpassung an eine Vielzahl unterschiedlicher Instrumentierungsaufgaben.

1. Vorbemerkungen

Beim Übergang zu intelligenten Systemkonzeptionen für die Prozeßsignalerfassung, -verarbeitung und -ausgabe sind insbesondere zwei Entwicklungsrichtungen zu unterscheiden /1/:

1. *Intelligente Analogwerterfassungssysteme,* die in der Regel in Form einer oder mehrerer Leiterkarten in Mikrorechner oder Personalcomputer (bzw. in ein an diese angeschlossenes externes Erweiterungsgerät) einsteckbar sind, und über einen parallelen Bus mit dem Rechner verbunden werden /2/, und
2. *Intelligente Meßumformer /3/ und Prozeßkoppelmodule* zur prozeßnahen dezentralen Prozeßgrößenerfassung und Prozeßbeeinflussung, die über einen seriellen digitalen Feldbus untereinander und mit einem Leitgerät, z. B. einem Personalcomputer – zunehmend über standardisierte Schnittstellen –, verbunden sind. Eine Auswirkung des Mikroprozessor- und Einchipmikrorechnereinsatzes in der Meß- und Automatisierungstechnik ist der allmähliche Übergang zu dezentralen prozeßnahen Systemkonzeptionen, wodurch sich eine völlig neue Generation von Meß- und Automatisierungskonzeptionen herausbildet. Hauptbestandteile solcher dezentralen Systeme sind die genannten intelligenten Module zur Meßgrößenerfassung und zur Beeinflussung des Prozesses. Die Datenvorverarbeitung und auch ein Teil der Datenverarbeitung erfolgen in direkter Sensornähe, wobei langfristig der Übergang zum intelligenten Sensor realisiert wird. Mit dieser Umwälzung verbunden ist der Übergang von den bisherigen analogen Systemkonzeptionen mit dem in der Prozeßautomatisierung langjährig eingeführten und weit verbreiteten 4 ... 20-mA-Stromsignal auf ein rein digitales serielles Feldsbussignal (Übergang von der analogen Sternstruktur zur digitalen Busstruktur) /4/.
Nachfolgend wird ein zur zweiten Gruppe gehöriger intelligenter prozeßnaher programmierbarer Prozeßkoppelmodul (PPM) mit seriellem Feldbusanschluß vorgestellt, der in erster Linie als intelligenter mehrkanaliger Meßumformer, darüber hinaus aber auch für Steueraufgaben (Binärwerterfassung, -verknüpfung und -ausgabe, Impulszählung, Zeitmessung), Grenzwertüberwachung, Regelung u. a. einsetzbar ist.

2. Gesamtkonzept

Ausgehend von praktischen Anforderungen wurde ein programmierbarer Prozeßkoppelmodul entwickelt, bei dem 7 Eingänge in beliebig gemischter Kombination ohne Hardwareänderung wahlweise als analoge Sensoreingänge (direkter Sensoranschluß) oder als Binäreingänge (Schalterabfrage, TTL-Pegel u. a.) nutzbar sind. Zusätzlich ist ein Zählein- und -ausgang verfügbar. Der Modul ist mit einem Analogausgang, einem von der übrigen Schaltung galvanisch getrennten seriellen Feldbusanschluß und vier Binärausgängen versehen. Letztere verleihen dem PPM Eigenschaften eines Stellmoduls (z. B. Alarmabgabe, speicherprogrammierbare Steuerung) bzw. eines kombinierten Meß- und Stellmoduls (Möglichkeit der Rückmeldung von Analog- und Binärsignalen nach Stellgliedverstellung, Rückmeldung von Endlagen, Stellgliedhavarien usw.). Auch der Einsatz im Inselbetrieb ist dadurch gegeben

(prozeßnahe Meßwert-/Binärwerterfassung mit direkter Rückwirkung auf den Prozeß).

Folgende *Haupteigenschaften* kennzeichnen den Anwendungsbereich:

- max. 7 analoge oder binäre Sensorsignale erfaßbar
- Auflösung der Analogsignalverarbeitung 12 Bit; Eingangsspannungsbereich 39 mV . . . 5 V (softwareprogrammierbar)
- ein Analogausgang im Bereich 0 . . . 20 mA bzw 0 . . . 10 V beliebig skalierbar
- ein Zählein- und -ausgang
- max. 4 binäre Signalausgänge (mit Treibern)
- Abtastrate ≈ 4000 1/s
- serielle Feldbusschnittstelle als Digitalausgang mit bidirektionaler Kommunikation, galvanisch getrennt
- max. 128 Funktionseinheiten (PPM) an einen Feldbus adressierbar
- Parameter (Meßbereichsanfang, -ende, Grenzwerte) und Konfiguration des Moduls wahlweise ferneinstellbar oder durch EPROM und Programmierfeld einstellbar
- Ferndiagnose [Zugriff auf die Register des Einchipmikrorechners (EMR) über Feldbus möglich]
- galvanische Trennung jeweils zwischen Analogteil mit EMR, Digitalbus und Hilfsenergieversorgung.



Prof. Dr. sc. techn. Manfred Seifart (56) studierte von 1951 bis 1956 an der TU Dresden Hochfrequenztechnik. Anschließend war er 13 Jahre im Wissenschaftlichen Industriebetrieb VEB Vakutronik Dresden (heute VEB Robotron-Meßelektronik „Otto Schön") auf dem Gebiet der Entwicklung kernphysikalischer und elektronischer Meßgeräte in verschiedenen Funktionen tätig, zuletzt als Entwicklungsleiter.
1969 erfolgte die Berufung an die TH Magdeburg auf den Lehrstuhl Elektronische Bauelemente und Schaltungstechnik.
Seit 1983 ist er Leiter des Wissenschaftsbereiches Prozeßmeßtechnik; Mitglied des Vorstandes der WS Computertechnik, Leiter des FA Mikroprozessor-Interfacesysteme.

Derzeitige Forschungsarbeiten: Intelligente Meßwerterfassung, Mikroprozessor-Interfacesysteme, Sensorsignalerfassung und -verarbeitung.

Der im PPM enthaltene Einchipmikrorechner ermöglicht eine Meßwertvorverarbeitung, die mit der Software leicht an spezielle Anwenderforderungen anpaßbar ist. Beispiele sind Routinen zur Linearisierung von Sensorkennlinien, Nullpunkt- und Driftkorrektur des Meßsystems, Mittelwertbildung, Aussondern verfälschter Meßwerte, Grenzwertüberwachung.

Der EMR ermöglicht in Verbindung mit dem EPROM folgende weitere flexible Eigenschaften:

– Anpassung an völlig neuartige Meßaufgaben durch Austausch des EPROMs
– Verarbeitungsroutinen wahlweise über Programmierfeld oder über das Funktionsfeld des Übertragungsprotokolls einstellbar (fernsteuerbar)
– Eigenüberwachung (Programmlauf, Betriebsspannung)
– Fehlermeldung

Über den digitalen Feldbus sind Ferneinstellung und Ferndiagnose von einem Steuerrechner bzw. Koppelrechner aus

Bild 1 Blockschaltbild des programmierbaren Prozeßkoppelmoduls

möglich. Gleichzeitig wird damit die Schnittstelle zur höheren Prozeßrechnerebene realisiert.

3. Schaltungsrealisierung (Bild 1)

Mit Hilfe eines Eingangsmultiplexers wird einer von den 7 Eingangskanälen angewählt und zum Meßverstärkereingang durchgeschaltet. Mit einem zweiten Multiplexer erfolgt in Verbindung mit einem Präzisionswiderstandsnetzwerk die automatische Verstärkungsumschaltung im Bereich von 2...256 (≙ Eingangsspannung 39 mV...5 V). Bei der gewählten Auflösung von 12 bit beträgt die kleinste nachweisbare Eingangsspannung 10 μV und liegt damit in der Größenordnung des Rauschens.
Der achte Eingang des Analogmultiplexers im Eingangskreis wird als Bezugspotential für die max. 7 Analogeingänge verwendet. Wegen der galvanisch getrennten Ankopplung an den digitalen Feldbus kann auf einen aufwendigen Instrumentationsverstärker verzichtet werden, da die Gleichtaktunterdrükkung durch die galvanische Trennung bewirkt wird. Grundsätzlich besteht die Möglichkeit, durch Zufügen eines weiteren Analogmultiplexers auch echten Differenzbetrieb am Eingang zu realisieren /5/.
Jeder Eingang ist mit einer Eingangsschutzbeschaltung versehen, die Überspannungen von den Multiplexern fernhält. Über einen dritten Analogmultiplexer wird ein von der Referenzquelle B 589 konstant gehaltener Speisestrom von 1 mA zur Speisung von Widerstandssensoren zur Verfügung gestellt. Diese Anordnung im Multiplexbetrieb ersetzt 7 getrennte Stromquellen, die beim Anschluß von 7 Widerstandssensoren erforderlich wären.
Die Eingangsschaltung ist extrem leistungsarm dimensioniert (22 mW). Beim Umschalten der einzelnen Eingangskanäle muß jeweils die Einschwingzeit des Meßverstärkers abgewartet werden, bevor der Meßwert weiterverarbeitet werden kann (AD-Umsetzung). Die Einschwingzeit ist verstärkungsabhängig. Unter Verwendung des leistungsarmen Operationsverstärkers B 176 mit $I_{Set} = 15\,\mu A$ beträgt die Einschwingzeit bei voller 12-Bit-Auflösung 4 μs bei V = 2, 128 μs bei V = 64 und 528 μs bei V = 256. Zu diesen Werten muß noch ein Betrag von 20 μs addiert werden, der durch die Slew-Rate des Operationsverstärkers bedingt ist. Grundsätzlich besteht die Möglichkeit, durch Zuschalten eines weiteren Operationsverstärkers die Einschwingzeit verstärkungsunabhängig auf etwa 5 μs zu reduzieren.
In Verbindung mit dem EMR erfolgt ein automatischer Offset- und Endwertabgleich, so daß Langzeitdriften praktisch völlig unwirksam werden.
Zur AD-Umsetzung wurde das Verfahren der sukzessiven Approximation verwendet. Zur Minimierung des Aufwands und des Leistungsverbrauchs wird ein 12-Bit-DAU C 565 eingesetzt, der sowohl für die AD- als auch für die DA-Umsetzung Verwendung findet. Die AD-Umsetzung wird realisiert, indem die Eingangsspannung in einem Analogkomparator mit der DAU-Ausgangsspannung verglichen wird. Der Algorithmus der sukzessiven Approximation wird vom EMR erzeugt. Die Umsetzzeit beträgt ≈260 μs. Während der AD-Umsetzung wird die analoge Ausgangsspannung in einer Abtast- und Halteschaltung gehalten.
Die Adresse des PPM ist mit einem DIL-Schalter in Verbindung mit einem Programmierfeld und einem Dekoder einstellbar. Das Programmierfeld ermöglicht dem Anwender, bestimmte Programmteile, Softwarebedingungen und Konfigurationen des PPM ohne Veränderung der Hard- und Software des Moduls auszuwählen und zu kombinieren.

4. Konstruktive Gestaltung

Die konstruktive Gestaltung des PPM wurde in Form einer Grundleiterkarte ($160 \times 120\,mm^2$) und einer Koppelkarte ($80 \times 120\,mm^2$) realisiert. Die Grundkarte ist für alle Anwendungsfälle gleich. Von der Koppelkarte können bei Bedarf mehrere Varianten realisiert werden, um die speziellen Forderungen der Sensorankopplung, der Binärein- und -ausgabe oder weitere Interfaceforderungen zu erfüllen (Kanalanzahl, Organisation der analogen und binären Ein-/Ausgänge).
Sensoren lassen sich wahlweise in 2-, 3- und 4-Leitertechnik anschalten. Infolge der Speisung von Widerstandssensoren mit dem Konstantstrom von 1 mA wirken sich auch bei größerer Entfernung zwischen dem Sensor und dem Verstärkereingang Spannungsabfälle auf den Leitungen nicht als Fehler aus.

5. Software

Der Umfang installierbarer Software und die damit verbundenen Möglichkeiten komfortabler Meßdatenverarbeitung werden im wesentlichen durch den verwendeten EMR U8820 bzw. U8840 mit seiner begrenzten Registeranzahl und dem verwendeten Programmspeicher (2 bzw. 4 Kbyte EPROM) bestimmt. Nähere Untersuchungen zeigten, daß für den Einsatz des PPM als Meßmodul die 2-Kbyte-EPROM-Variante ausreichend ist, falls keine umfangreichen Vorverarbeitungsaufgaben gelöst werden müssen. Für die zusätzliche Anwendung des PPM zur Regelung vor Ort ist es erforderlich, die Anzahl der Analogeingangskanäle auf etwa 4 zu beschränken (begrenzte RAM-Kapazität und Dynamik) /6/. Hauptforderungen an die Software sind Modularität, Universalität/Variabilität und leichte Programmierbarkeit vor Ort ohne Spezialkenntnisse. Da die letztgenannte Eigenschaft eine dominierende Rolle spielt, wurde das Interpreterprinzip angewendet. Das Programm besteht im wesentlichen aus den Modulen

- Initialisierungsmodul
- Prozeßkoppelmodul
- Buskommunikationsmodul

Die *Initialisierung* beinhaltet das Festlegen der Arbeitsweise der Tore, den EPROM- und RAM-Test, das Einlesen des Programmierfeldes und die Empfangsvorbereitung. Das Programmierfeld ermöglicht es, mittels Diodenbestückung die Anzahl der Binärkanäle und ihre Verknüpfung (Durchschalten; UND; ODER; EX-ODER; nicht Verarbeiten; Negieren; NAND; NOR; EX-NOR) festzulegen. Dadurch ist auch weitgehend die Softwarestruktur des Moduls bestimmt.
Der *Prozeßkoppelmodul* beinhaltet folgende Teilmodule:
- Analog- und Binärwertaufnahme
- automatische Verstärkungsänderung
- Nullpunkt- und Driftkorrektur der Meßkette
- Grenzwerttest als Sinnfälligkeitstest
- Normierung bzw. Skalierung der Meßwerte
- Störfilterung durch gleitende Mittelwertbildung
- Schrankentest.

Die Meßwertverarbeitung wurde zeitinterruptgesteuert konzipiert. Zum Erzielen einer hohen Zuverlässigkeit dienen Speicherschutzmaßnahmen (RAM- und EPROM-Test).

6. Serieller Bus

Jeder serielle Bus enthält die drei Hauptbestandteile Übertragungsmedium, serielle digitale Schnittstelle und Busprotokoll.
Als Übertragungsmedium wird eine verdrillte Zweidrahtleitung verwendet, gespeist mit 24 V Gleichspannung. Die Buskoppler der PPM enthalten eine Endstufe mit offenem Kollektor. Beim Senden einer Entstufe wird der Bus kurzgeschlossen (L-Pegel: 0 V; H-Pegel; 24 V). Die Datenübertragung über den Feldbus erfolgt asynchron im Manchesterkode (auch NRZ-Kode wählbar) nach einem PDV-Bus-ähnlichen Übertragungsprotokoll. Die Datentelegramme können unterschiedliche Länge

besitzen, die neben anderen spezifischen Informationen durch das Funktionsfeld bestimmt wird. Bei der Informationsübertragung folgen nach dem Adreßbyte das Funktionsbyte und die Datenbytes mit anschließender Datensicherung (CRC oder Parität).

7. Einsatzvorteile

Die universelle Konzeption des PPM ermöglicht einen sehr breiten Einsatzbereich. Die Flexibilität der Systemstruktur bringt wesentliche Vorteile bei der Projektierung und dem Betrieb in Automatisierungseinrichtungen wie z. B.
– drastische Kabeleinsparung durch Nutzung des digitalen Feldbusses nach dem Party-Line-Prinzip
– hohe Zuverlässigkeit der Meßdatenübertragung und hochgenaue Meßwertaufnahme durch Digitalisierung am Sensor
– gleiche Hardwarestruktur für unterschiedlichste Aufgaben
– Verknüpfung von Meßsignalen und Vorverarbeitung im PPM möglich (Beispiel: Wärmemengenmessung).
Die Leistungsaufnahme beträgt gegenwärtig 3,5 W. Bei stärkerem Einsatz von CMOS-Schaltkreisen läßt sich in Zukunft die Leistungsaufnahme beträchtlich reduzieren.

🖂 KONTAKT ☏

Technische Hochschule Magdeburg,
Sektion Technische Kybernetik und Elektrotechnik,
Postfach 124, Magdeburg, 3010,
Tel. 59 21 31 49, Dr. Beikirch

Literatur

/1/ Seifart, M.; Beikirch, H.; Baumecker, D.; Fincke, U.: Systeme zur Sensorsignalerfassung und -verarbeitung. Wiss. Z. Techn. Hochschule Magdeburg 30 (1986)

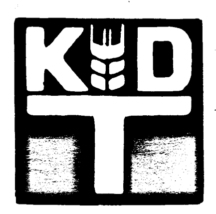
/2/ Seifart, M.; Hertwig, B.; Hanisch, H.: Intelligente Analogwerterfassungsplatine für das Mikrorechnersystem K 1520. Wiss. Z. Techn. Hochsch. Magdeburg 29 (1985) 7, S. 105

/3/ Seifart, M.; Beikirch, H.: Intelligenter Meßumformer. 30. Intern. wiss. Koll. TH Ilmenau 1985, Vortragsreihe „Prozeßmeß- und Sensortechnik"

/4/ Kriesel, W. u. a.: Neuartige Ansätze für die Technik künftiger Automatisierungsanlagen mit intelligenten Geräten. Vortrag zur 5. Wiss. Konferenz Anlagenautomatisierung. TH Leipzig 21. 5. 1986.

/5/ Dokumentation: Programmierbarer Prozeßkoppelmodul. Techn. Hochschule Magdeburg, Sektion TK/ET, 1986

/6/ Fincke, U.: Software für PPM. Diplomarbeit TH Magdeburg, Sektion TK/ET, 1985.



2. Zentrales Anwenderseminar Mikroelektronik des Präsidiums der KDT

Das im Mai 1987 stattfindende Anwenderseminar ist ein Bestandteil der KDT-Initiative „Spitzenleistungen für Schlüsseltechnologien". Es soll einen entscheidenden KDT-Beitrag leisten, um die Erfahrungen produktiver Anwendung der Mikroelektronik auszuwerten und zu verbreiten.

Schwerpunkte:

• Herausbildung von konkreten Aufgabenstellungen und von Wegen zur wirksameren Fortführung der sozialistischen Gemeinschaftsarbeit bei der Applikation Mikroelektronik über die zentralen Objekte der Fachverbände und andere KDT-Initiativen und Verpflichtungen.
• Mobilisierung einer noch größeren Zahl von KDT-Mitgliedern, insbesondere der jungen Wissenschaftler und Ingenieure, für Spitzenleistungen bei der Entwicklung, Produktion und Anwendung modernster elektronischer Technik.
• Auslösen neuer schöpferischer Initiativen der Entwickler und Konstrukteure bei der umfassenden Anwendung der Mikroelektronik zur Intensivierung.
• Vermittlung beispielgebender Erfahrungen der Betriebssektionen, Kombinatsaktive und Bezirksfachsektionen Elektrotechnik/Elektronik.
• Darlegung von Ergebnissen und Erkenntnissen zu Entwurf und Einsatz von semikundenspezifischen Schaltkreisen.
• Information und Beratung zur thematischen Entwicklung der KDT-Weiterbildung der ingenieurtechnischen Kader sowohl in bezug auf den konkreten Einsatz der Mikroelektronik und der rechentechnischen Mittel als auch in Hinsicht auf die vorlauforientierte Wissensvermittlung zur Anwendung der Schlüsseltechnologien.
Das 2. Zentrale Anwenderseminar wird an einem Tag in der letzten Maiwoche in Leipzig, Messehalle 7, mit einem Plenarvortrag des Präsidiums der KDT und anschließend in 5 Arbeitsgruppen durchgeführt. Der Termin ist so gewählt, daß zur gleichen Zeit in einer Halle des Messegeländes die wissenschaftlich-technische Angebotsmesse zum Thema „Nachnutzung rechnergestützter Rationalisierungslösungen, CAD/CAM" stattfindet.

Themen der Arbeitsgruppen

1. Erfahrungen aus der Anwendung von Rechner-, Steuer- und Leistungselektronik-Baugruppen für Automatisierungslösungen
2. Neue KDT-Initiativen zur Erweiterung des eigenständigen Entwurfs und der Fertigung anwenderspezifischer Schaltkreise
3. Orientierungen für die KDT-Objektbewegung zur Rationalisierung produktionsvorbereitender und der Verwaltungsprozesse durch die Nutzung rechentechnischer Mittel auf der Basis PC 1715
4. KDT-Aufgaben bei der Entwicklung, Vervollkommnung, Verwaltung und Bereitstellung von Software
5. KDT-Initiativen zur Vorbereitung und Einführung rechnergestützter Rationalisierungslösungen in Klein- und Mittelbetrieben.

KDT-Ideenwettbewerb für neue Konsumgüter

Die Preise für die besten Einsendungen zum Ideenwettbewerb der Kammer der Technik für neue elektrische und elektronische Konsumgüter sind Ende September in Berlin verliehen worden.

Der Ideenwettbewerb war am 1. Oktober 1985 in Vorbereitung des XI. Parteitages der SED vom Präsidium der Kammer der Technik ausgeschrieben worden. Thema waren u. a. Ideen und Lösungsvorschläge für neue Erzeugnisse für Haushalt, Wohnung sowie auf den Gebieten Unterhaltungselektronik und Computertechnik.

Insgesamt wurden 3366 Ideen, Lösungen und Vorschläge eingereicht, die von einer Jury aus Fachleuten staatlicher Institutionen, von Kombinaten und der Ingenieurorganisation bewertet wurden. In die engere Wahl kamen 2359 Ideen, während 1017 Einsendungen wegen mangelnden schöpferischen Gehalts ausschieden. Etwa 25 Prozent der ausgewählten Ideen regt die Entwicklung neuartiger Konsumgüter an, während die restlichen 75 Prozent Vorschläge zur Weiterentwicklung bereits existierender, gefragter industrieller Konsumgüter unterbreiten. Rund 40 Einsendungen sind mit Patenten oder Schutzrechts- und Patentanmeldungen belegt. Die Ideen werden zur Realisierung ausgewählten Kombinaten und Betrieben angeboten, die über entsprechende technische und technologische Voraussetzungen verfügen. Beim Präsidium der KDT sind bereits Anfragen nach Vorschlägen zur Nutzung eingegangen. Die besten Lösungsvorschläge werden für Betriebe und Kombinate abrufbereit in einer zentralen Ideenbank Konsumgüter gespeichert.

U. a. wurden folgende Vorschläge ausgezeichnet:
grafikfähiger Heimcomputer, flexibler bauelementearmer Kleincomputer und ein Experimentier-Computerspielsystem (U 880).

Programmierung in C

Dr. Thomas Horn
Ingenieurhochschule Dresden,
Sektion Informationsverarbeitung

Die Programmiersprache C stellt eine moderne höhere Programmiersprache dar, die auf Grund ihrer Portabilität, Flexibilität und Effektivität, insbesondere auf 16- und 32-Bit-Rechnern, aber auch auf 8-Bit-Rechner heute eine weite Verbreitung gefunden hat. Ihr Einsatz ist nicht auf bestimmte Anwendungsgebiete begrenzt,aber durch die Möglichkeit einer maschinennahen Programmierung und hohen Laufzeiteffizienz konnte in der Systemprogrammierung die bislang dominierende Makroassemblerprogrammierung weitgehend reduziert werden. Die Systemprogrammierung ist damit ein Haupteinsatzgebiet der Programmiersprache C und begründet ihre besondere Bedeutung. Die mit diesem Beitrag eingeleitete Artikelreihe soll deshalb dem Leser eine Einführung in die Anwendung der Programmiersprache C geben.

0. Einleitung

C ist eine moderne universelle höhere Programmiersprache /1, 2/, die Anfang der siebziger Jahre von Dennis Ritchie für die Implementierung des Betriebssystems UNIX[1] auf Rechenanlagen vom Typ PDP-11 entwickelt wurde. Der für die Übersetzung der C-Programme benötigte Compiler war unter dem Betriebssystem UNIX selbst implementiert. UNIX /3/, größtenteils in C geschrieben, umfaßte etwa 13000 Quellzeilen in C und nur etwa 800 Quellzeilen in der Makroassemblersprache zur Anpassung an die Hardware, wie E/A-Driver, Interruptservice und Speicherverwaltung. Auf Grund der guten Portabilität der Programmiersprache C und der günstigen Eigenschaften von UNIX für die Softwareentwicklung ist UNIX Mitte der siebziger Jahre mit relativ wenig Aufwand auf verschiedene Groß- und Kleinrechner übertragen worden, wie IBM/360, IBM/370, Honeywell 6000, Interdata 8/32, VAX-11/780 u. a. /4, 5/. Gegenwärtig ist UNIX auf mehr als 100000 Rechenanlagen vorwiegend in der Softwareentwicklung, Forschung und Ausbildung im Einsatz.

1 Eingetragenes Warenzeichen der Bell Laboratories.

Aus der Literatur ist zu ersehen, daß UNIX mit der Entwicklung der 16-Bit-Mikroprozessoren eine starke Verbreitung als „Standard"-Betriebssystem für die 16- und 32-Bit-Mikroprozessortechnik erfahren hat /6, 7/. Infolgedessen fand auch C eine weite Verbreitung. Da die Programmiersprache aber nicht auf das Betriebssystem UNIX oder auf bestimmte Anwendungsgebiete spezialisiert ist, sondern sich durch hohe Flexibilität, Kompaktheit, Effektivität sowie moderne Konzepte für den Steuerungsfluß und für die Arbeit mit Datenstrukturen auszeichnet, ist sie insbesondere in der Systemprogrammierung als echte Alternative zur Makroassemblersprache zu betrachten. Da wie bei allen höheren Programmiersprachen die Anwendung von C nicht auf UNIX beschränkt ist, sind in den letzten Jahren zahlreiche Implementierungen von C-Compilern unter vielen Betriebssystemen auf fast allen Rechnern bekannt geworden.
Die Anpassung der Programmiersprache C an verschiedene Betriebssysteme ist vor allem deshalb relativ problemlos, da C im Vergleich zu anderen Programmiersprachen keine READ- und WRITE- bzw. GET- und PUT-Statements enthält und somit frei von Besonderheiten der E/A- und Filesteuerung ist. Der Zugriff auf die E/A- und Filesteuerung sowie auf andere Systemdienste erfolgt grundsätzlich über Funktionsaufrufe (Prozeduren), was eine außerordentlich hohe Portabilität der C-Programme bei gleichzeitig guten Laufzeiteigenschaften sichert.
Die Besonderheiten der Gerätetechnik und des Betriebssystems finden somit ihren Niederschlag nur in der Funktionsbibliothek, die auch das C-Laufzeitsystem enthält und für jeden Compiler unter den einzelnen Betriebssystemen vorhanden sein muß. Praktische Untersuchungen zeigen, daß C-Programme in der Laufzeit mit Makroassemblerprogrammen guter bis sehr guter Programmierer in etwa vergleichbar sind. Vom Speicherplatzbedarf ergeben sich bei den übersetzten C-Programmen durch das Einbinden des C-Laufzeitsystems Nachteile, die in Abhängigkeit von der Programmgröße und der Größe der Datenbereiche sowie in Abhängigkeit vom Betriebssystem im Mittel eine Vergrößerung des Arbeitsspeichers um 20...50 % erforderlich machen.

1. Einführung in die Lexik

Das Alphabet der Programmiersprache C umfaßt den vollständigen Zeichensatz des ASCII-Kodes einschließlich aller Sonderzeichen und der Kleinbuchstaben (96 Zeichen). Darauf aufbauend werden in C sechs Kategorien von Begriffen unterschieden:
- *Identifikatioren*
- *Trennzeichen*
- *Schlüsselwörter*
- *Konstanten*
- *Zeichenketten und*
- *Operatoren.*

1.1. Identifikatoren

Ein *Identifikator* (Symbol) ist eine Folge von Buchstaben und Ziffern, wobei das erste Zeichen ein Buchstabe sein muß. Die ersten 8 Zeichen sind signifikant. Klein- und Großbuchstaben sind verschiedene Zeichen. Das Unterstreichungszeichen (_) zählt als Buchstabe. Externe Identifikatoren werden durch das Basis-Betriebssystem in der Länge oft eingeschränkt. Im Betriebssystem OS-RW auf Rechenanlagen des SKR sind externe Identifikatoren z. B. auf 6 Zeichen beschränkt, wobei vom Taskbuilder (Linker) auch keine Klein- und Großbuchstaben unterschieden werden.

Beispiele:
n1, block, satz_1, ALPHA, N1, _file, alpha1000, alpha1001. Die Identifikatoren **n1** und **N1** sind verschieden, während **alpha1000** und **alpha1001** identisch sind, da sie in den ersten 8 Zeichen übereinstimmen.

1.2. Trennzeichen und Kommentare

Als *Trennzeichen* werden Leerzeichen (SP), Tabulatoren (HT), Neue Zeile (NL) und Kommentare gewertet. Sie dürfen zwischen den Begriffen beliebig stehen und werden bei der Syntaxanalyse ignoriert. Falls kein anderes Sonderzeichen vorhanden ist, dienen sie nur zur Trennung der Begriffe, wie Schlüsselwörter, Identifikatoren und Konstanten.
Kommentare werden durch die Zeichenfolge /* eingeleitet und durch */ beendet. Sie dürfen an beliebiger Stelle innerhalb der Quellzeile zwischen den Begriffen an Stelle eines Trennzeichens

stehen. Eine Schachtelung von Kommentaren ist nicht zulässig. Kommentare werden im Programm zur Erläuterung der Programmanweisungen *(statements)* und somit zur Erhöhung der Lesbarkeit des Programms verwendet.

Beispiel:
int/*SPEZIFIKATION
VON I-VARIABLEN*/L,L1,L2;
L=25; /*ANFANGSWERT SETZEN*/

1.3. Schlüsselwörter

Folgende Identifikatoren sind als *Schlüsselwörter* reserviert:

asm	enum	short
auto	extern	sizeof
break	float	static
case	for	struct
char	fortran	switch
continue	goto	typedef
default	if	union
do	int	unsigned
double	long	void
else	register	while
entry	return	

Diese Schlüsselwörter dürfen anderweitig nicht noch einmal vergeben werden. Das Schlüsselwort **entry** ist gegenwärtig i.allg. noch nicht implementiert, aber für spätere Erweiterungen reserviert. In modernen C-Compilern sind auch die Schlüsselwörter **void**, **enum**, **asm** und **fortran** implementiert oder reserviert. Die Anwendung der aufgeführten Schlüsselwörter wird in den weiteren Abschnitten erläutert.

1.4. Konstanten und Zeichenketten

In C werden folgende Konstantentypen unterschieden:
- *Integerkonstanten*
- *Gleitkommakonstanten*
- *ASCII-Zeichenkonstanten und*
- *ASCII-Zeichenkettenkonstanten.*

Integerkonstanten

Integerkonstanten sind standardgemäß Dezimalzahlen. Eine führende **0** kennzeichnet Oktalzahlen, während Hexadezimalzahlen mit der Zeichenkombination **0x** bzw. **0X** beginnen. Als Hexadeziмalziffern sind neben den Ziffernzeichen die Buchstaben **A** bis **F** bzw. **a** bis **f** vorgesehen. Die Zahl **200** kann somit als **200**, **0310** oder **0xc8** dargestellt werden.
Integerkonstanten sind standardgemäß vom Datentyp **int** ,das heißt, sie werden auf den meisten Klein- und Mikrorechnern intern im 16-Bit-Format dargestellt. Eine Dezimalkonstante, deren Wert mit Vorzeichen 16 Bit überschreitet, wird als Konstante vom Datentyp **long** (32 Bit) vereinbart. Das gleiche trifft für Oktal- und Hexadezimalzahlen zu, die ohne Vorzeichen 16 Bit überschreiten. Integerkonstanten können durch einen nachgestellten Buchstaben **l** bzw. **L** explizit im long-Format vereinbart werden.

Beispiele:
XC2FF,0177000, −1000, 0777000, 0XffcOL, +200l.

Die ersten drei Zahlen werden im 16-Bit-Format und die letzten drei Zahlen im 32-Bit-Format gespeichert.

Gleitkommakonstanten

Gleitkommakonstanten bestehen aus einem ganzzahligen Teil, dem Dezimalpunkt, einem Bruchteil, einem Buchstaben **e** oder **E** und einer Integerkonstanten (wahlweise mit Vorzeichen). Der ganzzahlige Teil oder der Bruchteil (aber nicht beide zugleich) sowie der Exponententeil (**e** bzw. **E** mit Integerkonstante) können wahlweise entfallen. Wenn ein Exponent angegeben wird, kann die Angabe des Dezimalpunktes entfallen. Gleitkommazahlen werden intern generell im Format mit höherer Genauigkeit (64 Bit) dargestellt.

Beispiele:
2E3 , 2.0 , .518 , 3.51e−10 , −.5e+3

ASCII-Zeichenkonstanten

Eine Zeichenkonstante wird in Hochkommas, z.B. **'a'** , eingeschlossen. Ihr numerischer Wert entspricht dem Wert des Zeichens in der ASCII-Tabelle. Nichtdruckbare Zeichen, das Hochkomma (') selbst und der Backslash (\) werden mittels Escape-Folgen dargestellt, die durch den Backslash eingeleitet werden:

BS	Backspace (Rücksetzen um ein Zeichen):	\b
CR	Carriage return (Wagenrücklauf):	\r
FF	Form feed (Formularvorschub):	\f
HT	Horizontal tabulator (Tabulatorsprung):	\t
NL(LF)	Newline (Neue Zeile/ Zeilenvorschub):	\n
NUL	Null (kein Zeichen):	\0
\	Backslash (linksgeneigter Schrägstrich):	\\
'	Single quote (Hochkomma):	\'

Alle anderen nichtdruckbaren Zeichen (Steuerzeichen) werden im Oktalkode durch die Escape-Folge *ddd* vereinbart, wobei *ddd* bis zu drei Oktalziffern verkörpert. Wenn nach einem Backslash kein gültiges Zeichen folgt, so wird der Backslash ignoriert.

Beispiele: 'a', '0', 'Z', '\0', '\n', '\'', '\\', '\33', '\177'

Die Konstante '0' entspricht dem ASCII-Zeichen Null und definiert ein Byte mit dem Wert oktal 60, während '\0' ein Byte mit dem Wert 0 definiert.

ASCII-Zeichenkettenkonstanten

Eine Zeichenkettenkonstante ist eine Folge von ASCII-Zeichen, die in Anführungszeichen eingeschlossen ist, z.B. **„abcdef"**. Vom Compiler wird die Zeichenkette mit einem Null-Byte \0 abgeschlossen (Endekennzeichen).
In Zeichenketten sind die schon beschriebenen Escape-Folgen für die Darstellung der Sonder- und Steuerzeichen zulässig (siehe Zeichenkonstanten). Ein Anführungszeichen in einer Zeichenkette muß durch die Escape-Folge \" dargestellt werden. Ein Backslash mit einem sofort folgenden Newline wird ignoriert und kann verwendet werden, um eine Zeichenkettenkonstante über mehrere Zeilen zu schreiben. Eine Zeichenkettenkonstante wird vom Datentyp „Zeichenfeld" mit der Speicherklasse **static** vereinbart, das mit den ASCII-Kodes der vorgegebenen Zeichenkette initialisiert ist. Alle Zeichenkettenkonstanten, auch wenn identisch geschrieben, werden vom Compiler getrennt gespeichert.

Beispiele:

„Das ist ein Beispiel. \n",
„Heute ist \"Montag\".\f"

1.5. Operatoren und Operanden

In C sind einstellige Operatoren, zweistellige Operatoren und Zuweisungsoperatoren vorhanden.
Einstellige Operatoren stehen in der Regel vor einem Operanden, in Ausnahmefällen auch dahinter. Zweistellige Operatoren verbinden zwei Operanden über eine arithmetische oder logische Operation. Zuweisungsoperatoren werden für die Zuweisung eines Wertes einer Variablen genutzt.
Operanden können Konstanten, Identifikatoren und aus ihnen aufgebaute Ausdrücke sein (siehe Pkt. 3.).
Ein Identifikator bezeichnet einen bestimmten Speicherbereich und wird

durch zwei Attribute charakterisiert: den Datentyp und die Speicherklasse, die im nachfolgenden Abschnitt erläutert werden. Der Speicherbereich, der an den Identifikator gekoppelt ist, wird im allgemeinen als Objekt bezeichnet. Der Datentyp legt die Bedeutung des Wertes des Objekts fest, während die Speicherklasse die Adresse und das Zeitverhalten des Objekts bestimmt.
Wie wir später sehen werden, kann man in C unter Ausnutzung des Zeigerkonzeptes (pointer) mit den Adressen von Objekten wie in der Assemblersprache arbeiten. Die Adresse eines Objekts wird über den Adreßoperator & gebildet, z.B. **&VAR** – Adresse der Variablen **VAR**. **&VAR** ist somit Ausdruck vom Zeigertyp (pointer). Der Indirektoperator * dagegen dient dem Zugriff auf die Objekte über ihre Adressen bzw. über Adreßausdrücke. Angenommen **E=&VAR**, dann muß E eine Variable vom sogenannten Zeigertyp sein, und über * E kann auf die Variable **VAR** zugegriffen werden. Es sind somit die beiden folgenden Schreibweisen identisch:

```
VAR=0
*E=0 .
```

Beide Schreibweisen zeigen, daß links vom Zuweisungsoperator (linker Teil der Ergibtanweisung) entweder einfache Variablen oder Adreßausdrücke stehen dürfen, die auf eine einfache Variable über den Indirektoperator verweisen. Wie in allen anderen Programmiersprachen dürfen rechts vom Zuweisungsoperator selbstverständlich beliebige Ausdrücke stehen.

1.6 Der Programmaufbau

Ein C-Programm ist für den Compiler eine beliebig lange Programmzeile, die aufgrund der begrenzten Zeilenlänge bei Bildschirmgeräten und Druckern beliebig in Zeilen kürzerer Länge untergliedert werden darf. C ist analog den ALGOL-ähnlichen Sprachen eine blockstrukturierte Sprache, wobei allerdings zur Erhöhung der Kompaktheit des Programms die BEGIN- und END-Klammern im Schriftbild durch geschweifte Klammern ersetzt werden. In C ist jedes Programm grundsätzlich eine Funktion, die aus dem Funktionskopf und dem Funktionsblock besteht. Funktionen sind etwa den Begriffen SUBROUTINE und FUNCTION bei FORTRAN äquivalent, unabhängig davon, ob ein Funktionswert im konkreten Fall als Ergebnis übergeben wird oder nicht.
Jedes Programm ist daher in C eine Funktion und kann als ein Baustein von anderen Funktionen aufgerufen werden. Eine Funktion, die vom Betriebssystem gestartet werden soll, muß immer den Namen **main** tragen und kann Parameter aus dem Startkommando übernehmen. Alle anderen Funktionen können beliebige Namen tragen und übernehmen ihre Parameter von der aufrufenden Funktion. Eine Funktion hat folgenden prinzipiellen Aufbau:

typ funktionsname (liste_der_formalen_parameter)
deklaration_der_formalen_parameter
{
deklarationen ;
anweisungen ;
}

wobei *typ* den Datentyp des Funktionswertes festlegt. Im Funktionsblock können lokale Variablen deklariert werden, nach deren Deklaration dann die Anweisungen folgen. In C werden alle Deklarationen und Anweisungen mit einem Semikolon (;) abgeschlossen. Ausführlich werden die Funktionen später behandelt.

Beispiel:

```
main ()
{
printf(„Hier ist unser
erstes Programm!\n");
}
```

Die Funktion **main** verfügt über keine formalen Parameter und keine lokalen Variablen. Die Funktion **printf** wird zum Ausdrucken einer Zeichenkette benutzt. Das abschließende Zeichen NL (\n) ist sehr wichtig, da es den Terminalpuffer leert und erst die Ausgabe der Zeichenkette bewirkt. Es sollte deshalb auch in den weiteren Beispielen nie vergessen werden!
Ausführlich wird die **printf**-Funktion später behandelt.

2. Einfache Datentypen und Speicherklassen

In C herrscht – ähnlich wie in ALGOL und PASCAL – ein Deklarationszwang, das heißt, alle Identifikatoren für Variablen müssen mit einem Typbezeichner und einem Speicherklassenbezeichner zu Beginn eines Blockes vereinbart werden.

2.1. Die Typbezeichner

Typbezeichner und ihre Interpretation sind:

char	für 8-Bit-Zeichen (Byte)
int oder short int	für 16-Bit-Worte mit Vorzeichen
long oder long int	für 32-Bit-Worte mit Vorzeichen
unsigned oder unsigned int	für 16-Bit-Worte ohne Vorzeichen
float	für 32-Bit-Gleitkommazahlen
double oder long float	für 64-Bit-Gleitkommazahlen

Beispiele:

```
char c1,c2;
long a,b,c;
lond float z0, z1,z2;
```

Objekte vom Datentyp **char** dienen zum Speichern von Zeichenketten. Ein Zeichen wird durch einen Integer-8-Bit-Kode (Byte) entsprechend der ASCII-Kodetabelle dargestellt.
Die Datentypen **short**, **int** und **long** bezeichnen Objekte vom Typ integer, wobei **short** und **int** dem 16-Bit-Format und **long** dem 32-Bit-Format entsprechen.
Der Datentyp **unsigned** bezeichnet vorzeichenlose Integerzahlen (16 Bit), die der modulo-65536-Arithmetik unterliegen. (Der Datentyp **unsigned long** ist nicht realisiert!) Der Datentyp **unsigned** wird insbesondere für die Speicherung von Adressen benutzt.
Die Datentypen **char** und alle Arten von **int** werden kurz als Integertypen bezeichnet.
Die Datentypen **float** und **double** realisieren Gleitkommazahlen im 32-Bit- bzw. 64-Bit-Format. Sie werden kurz als Gleitkommatypen bezeichnet.
Diese sechs genannten Datentypen sind die grundlegenden arithmetischen Typen (einfache Typen), wobei **char** als arithmetischer Typ im Byteformat betrachtet wird. Aufbauend auf die genannten einfachen Datentypen sind folgende höhere Typen möglich:

– Felder (*arrays*) von Objekten der Datentypen,
– Funktionen (**functions**),die ein Objekt mit einem vorgegebenen Datentyp zurückgeben,
– Zeiger (*pointers*) zu Objekten eines vorgegebenen Datentyps,
– Strukturen (*stuctures*), die eine Folge von Objekten verschiedener Datentypen enthalten, und
– Vereinigungen (*unions*),die verschiedenen Objekten verschiedener Datentypen den gleichen Speicherplatz zuweisen.

Die höheren Typen werden in den weiteren Abschnitten behandelt.

2.2. Definition neuer Typbezeichner

Mit dem Schlüsselwort **typedef** lassen sich neue Typbezeichner definieren. Sie können als Synonyme für andere Typ-

bezeichner oder für Strukturen bzw. Vereinigungen benutzt werden.

```
Beispiel:
typedef int tabelle, *index;
tabelle a [100], b [20];
index ai, bi;
```

Es wurden zwei neue Typbezeichner – **tabelle** und **index** – definiert, wobei **tabelle** dem Typ **int** und **index** einem Integer-Zeigertyp entsprechen. Die Felder und Zeigertypen werden erst später behandelt. Das Beispiel zeigt aber, daß **tabelle** und **index** jetzt neue Schlüsselwörter sind.

2.3. Die Speicherklassenbezeichner

Jede Variable ist in C an einen bestimmten Speicherklassenbezeichner gebunden, der die Art und Weise der Speicherplatzzuweisung für die Variablen festlegt. Vier Speicherklassenbezeichner sind vorhanden:

auto
static
extern
register

- Automatische Variablen (**auto**) sind lokale Variablen eines Blockes. Bei jedem Eintritt in einen Block wird ihnen dynamisch zur Laufzeit Speicher zugewiesen. Beim Verlassen des Blockes wird der Speicher wieder freigegeben.
- Statische Variablen (**static**) sind ebenfalls lokale Variablen eines Blokkes. Der Speicherplatz wird ihnen aber statisch zur Übersetzungszeit zugewiesen, so daß sie ihren Wert auch nach dem Austritt aus dem Block nicht verlieren. Statische Variablen, die außerhalb von Funktionen vereinbart werden, haben für alle Funktionen innerhalb eines Übersetzungsmoduls Gültigkeit.
- Externe Variablen existieren während der gesamten Programmausführung und können zum Datenaustausch zwischen den Funktionen benutzt werden. Sie müssen außerhalb von Funktionen in genau einem Übersetzungsmodul ohne Angabe eines Speicherklassenbezeichners global definiert werden. Die Bezugnahme auf global definierte Symbole erfolgt in den anderen Übersetzungsmoduln durch die Spezifikation des Speicherklassenbezeichners **extern** bei der Vereinbarung der entsprechenden Variablen. Dadurch wird ihnen kein neuer Speicherplatz zugewiesen, sondern sie werden der bereits global definierten Variablen gleichgesetzt.
- Registervariablen werden, wenn es möglich ist, in den „schnellen" Registern des Prozessors gespeichert. Registervariablen, denen kein freies Prozessorregister zugewiesen werden kann, werden wie automatische Variablen behandelt. Als Registervariablen sind nur Variablen vom Typ **int**, **unsigned**, **char** und *pointer* zulässig.

Wenn kein Speicherklassenbezeichner angegeben wird, gilt für alle Variablen innerhalb einer Funktion **auto** und außerhalb **static**.
Der Speicherklassenbezeichner wird dem Typbezeichner immer vorangestellt, zum Beispiel
static int a,b,c;
Bei dem Speicherklassenbezeichner **register** kann der Typbezeichner **int** ausgelassen werden.

2.4. Der Gültigkeitsbereich von Variablen

Variablen sind grundsätzlich immer in dem Block gültig, in dem sie definiert wurden. Bei ineinander geschachtelten Blöcken[2] sind sie auch für die inneren Blöcke gültig. Werden in einem inneren Block Variablen gleichen Namens wie im äußeren Block definiert, so gilt im inneren Block die Variable, die im inneren Block definiert wurde und im äußeren Block die im äußeren Block definierte Variable. Nur Variablen, die zu Beginn des Übersetzungsmoduls vor der ersten Funktion spezifiziert wurden, gelten für alle Funktionen des Übersetzungsmoduls ohne nochmalige Deklaration. Wurde bei solchen Variablen der Speicherklassenbezeichner **static** ausgelassen, so tragen sie zusätzlich das Attribut global. Auf globale Variablen kann von getrennt übersetzten Moduln durch Spezifikation des Speicherklassenbezeichners **extern** zugegriffen werden.

```
Beispiel:
int i;  /* Globale Variable i*/
main()
{
int j,k;  /* Automatische Variablen j
und k*/
i=10;
j=i;
incr();  /* Aufruf der Funktion incr*/
k=j+i;
printf("k=%d,j=%d  n",k,j);
}
incr()
{
extern int i;  /* Externe Variable i*/
i=i+1;
}
```

In der Funktion **printf** ist **%d** eine Formatspezifikation zum Einfügen einer Dezimalzahl. Für jedes **%d** wird somit der Wert des nächsten Parameters der Parameterliste substituiert. Als Ergebnis wird folgende Zeile auf dem Bildschirm ausgegeben:

k=21,j=10.

Literatur

/1/ Kernighan, B. W.; Ritchie, D. M.: The C Programming Language. Prentice-Hall, Inc., Englewood Cliffs, New Yersey, 1978
/2/ Kernighan, B. W.; Ritchie, D. M.: Programmieren in C. Carl-Hanser-Verlag, München, Wien, 1983
/3/ The Bell System Technical Journal. Vol. 57, Nr. 6, Part 2, (Juli – August 1978)
/4/ Lious, J.: Experience with the UNIX Time-Sharing System. Software – Practice and Experience, Vol. 9, 701–709 (1979)
/5/ Mitze, R.W.: The UNIX Systems as a Software Engineering Environments. In: Software Engineering Environments, H. Hühnke (Hrsg.), North-Holland, 1981
/6/ Lichte, I.; Harbers, H.: UNIX unterstützt Mikrocomputer-Entwicklungssystem. Elektronik, Heft 26/1982, S. 55–58
/7/ Wossidlo, M.: TNIX – eine UNIX-Implementierung für Entwicklungsaufgaben. Elektronik, Heft 26/1982, S. 61–65
/8/ Horn, Th.: Zur Anwendung der Programmiersprache C in der Systemprogrammierung. Wiss. Beiträge der IH Dresden, Heft 4, 1983, 30–42
/9/ Horn, Th.: Programmiersprachen – ein Vergleich an Hand von Beispielen. In: Kleinstrechner-TIPS, Fachbuchverlag Leipzig, 1986, 4–34

Vorschau:

Nachdem in diesem Teil der Artikelreihe zu C eine Einführung in die Lexik gegeben und die grundsätzlichen einfachen Datentypen und Speicherklassen behandelt wurden, werden im nächsten Heft die Operatoren und Ausdrücke sowie die Steuerstrukturen erläutert. Sowohl die Operatoren und Ausdrücke wie auch die Steuerstrukturen widerspiegeln maßgeblich das moderne und leistungsfähige Sprachkonzept der Programmiersprache C, das zu ihrer weiten Verbreitung in der Systemprogrammierung beigetragen hat.

---

2 Gemäß Sprachdefinition /1,2/ sind ineinander geschachtelte Blöcke zugelassen. Aber eine Reihe gegenwärtig weit verbreiteter Compiler unterstützt nur die Deklaration von Variablen zu Beginn des äußersten Funktionsblockes.

Semigrafik für PC 1715

Dr. Dieter Herden, Rolf Lüdicke,
Claus Wippich, Technische Hochschule
Leipzig, Sektion Elektroenergieanlagen

Der Beitrag beschäftigt sich mit der Möglichkeit einer semigrafischen Informationsverarbeitung auf dem PC 1715, die über eine hardwaremäßige Nachrüstung (Option) mit einem RAM-Baustein und einer entsprechenden Grafikgrundsoftware für vielfältige Anwendungsbereiche gewährleistet wird.
Unter Semigrafik wird eine Bildgenerierungmethode verstanden, die bei zeichenorientierter Addressierungsweise des Monitors unter Verwendung von n zusätzlichen Zeichengeneratoren die Qualität einer vollgrafischen Darstellung bei hoher Bildgenerierungsgeschwindigkeit ermöglicht.
Einzelheiten über den Aufbau, die Funktion und die Anwendungsgebiete des Mikrorechners PC 1715 werden in /1/ ausführlich dargestellt.

1. Hardwarekonzept

Die hardwaretechnische Lösung zeigt Bild 1 und wird in /2/ detailliert erläutert.
Bezogen auf die Standardausführung des PC 1715, stellt der Grafikbaustein eine nachrüstbare Leiterplatte dar, die ohne Veränderung des Gerätestandards im Steuerteil des Rechners befestigt und über zwei Steckverbinder angeschlossen wird. Dieser Grafikbaustein wird in einer Nutzerdokumentation /3/ beschrieben; die Bestückung ist mit handelsüblichen DDR-Bauelementen möglich.

2. Softwarekonzept

Das Softwarepaket GEDIT wurde in der Form eines allgemeinen Grafikeditors entwickelt. Analog zum Textverarbeitungsprogramm, das dem Anwender ein wahlfreies und komfortables Editieren beliebiger Texte ermöglicht, besteht das Anliegen von GEDIT darin, auf Zeichen-, Symbol- und Bildebene einen leicht handhabbaren und leistungsfähigen Funktionsvorrat zum Erstellen, Manipulieren, Verwalten, Speichern und Ausgeben von Grafiken zur Verfügung zu stellen.
Im Bild 2 werden, bezogen auf die grafische Informationsverarbeitung, die Grundfunktionen Editieren und Generieren gegenübergestellt.
Das Ziel dieser Funktionen besteht in der Bilderstellung durch Manipulation der Elemente Bildpunkt (Pixel), Zeichen (als geordnete Menge von Bildpunkten) und Symbol (als geordnete Menge von Zeichen). Typisch für den semigrafischen Verarbeitungsmodus ist das Editieren von Zeichen, da zumeist hardwarebedingt nicht der Bildpunkt, sondern das Zeichen als frei addressierbares Element auf dem Monitor in Erscheinung tritt. In unserem Fall besteht das über den ASCII-Code anwählbare Zeichen eines Zeichengeneratorfiles von 128 Zeichen aus $8*12$ Bildpunkten. Als Symbol wurde ein Zeichenverbund von $1 \leq n \leq 160$ Zeichen festgelegt. Symbole sind über ihren Namen erreichbar und werden in einer Symboldatei abgelegt.
Neben dem Editieren ist das Generieren einer Grafik über Programmprozeduren möglich, in denen Bildpunkt, Zeichen und Symbol gleichermaßen zur Bildgenerierung verwendet werden. Auf semigrafischer Grundlage ist das Generieren einer Grafik möglich und wird auch beim Erstellen von Symbolgrafiken bzw. statistischen Darstellungen mit hoher Effektivität angewandt. Dabei bestehen jedoch Probleme bei der Darstellung maßstäblicher Grafiken auf dem Monitor sowie bei Grafiken, deren Generierung die Anzahl der Zeichen des Zeichengenerators übersteigt.
Aus der Kombination der Grundverfahren Editieren und Generieren sind die im Bild 3 dargestellten Softwareschichten des Grundsoftwarepaketes GEDIT ableitbar.

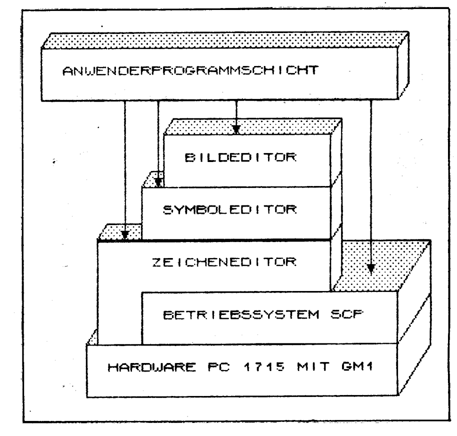


Bild 3 Schichtenstruktur des Grundsoftwarepakets GEDIT

3. Funktionsbeschreibung der implementierten Softwarelösung

Das Softwarepaket GEDIT stellt eine Grundsoftware dar, die dem Anwender auf den einzelnen Programmschichten Werkzeuge (Tools) zur interaktiven Generierung von

- Zeichen (Zeicheneditor)
- Symbolen (Symboleditor)
- Bildern (Bildeditor)

sowie zu deren Verwaltung, Abspeicherung und Ausgabe (Hardcopy, Formatausdruck) zur Verfügung stellt.
GEDIT wurde in Turbo-Pascal programmiert, um hohe Interaktivität, erforderliche Modularität, kurze Programmlaufzeiten und notwendige Portabilität auf nachfolgenden Rechnersystemen zu gewährleisten.

3.1. Anwendung der einzelnen Editoren

Der Zeicheneditor ermöglicht dem Anwender das interaktive Erstellen von n Zeichengeneratoren entsprechend den zu bearbeitenden Aufgabenstellungen. Zu diesem Zweck wird links auf dem Monitor ein Arbeitsfeld von $8*12$ wahl-

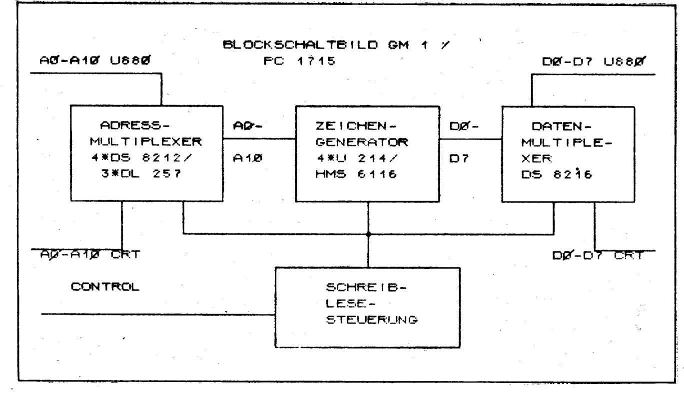


Bild 1 Funktionsschema des Grafikmoduls GM 1

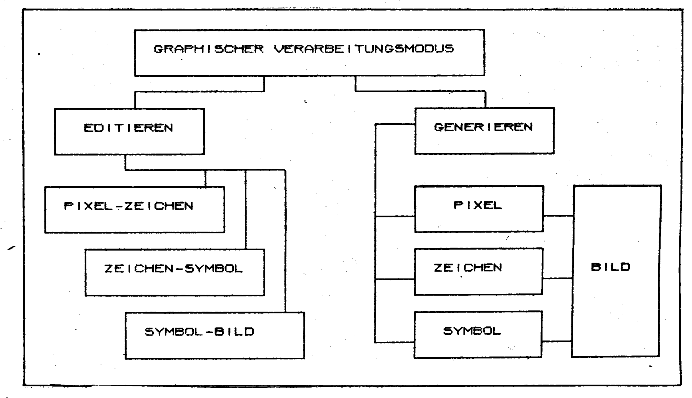


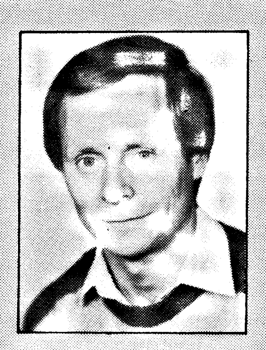
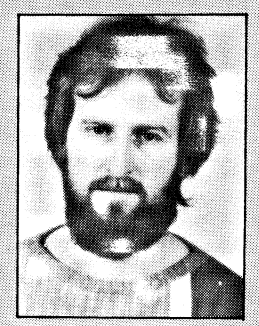
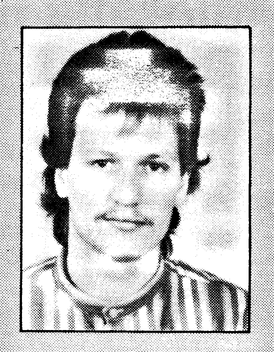
Bild 2 Grafische Grundfunktionen

frei belegbaren Bildpunkten zur Zeichengenerierung in vergrößerter Darstellung und rechts der aktuelle Inhalt des Zeichengeneratorfiles in Originalgröße dargestellt (Bild 4).
Bild 5 zeigt das Arbeits- und Menüfeld zur Editierung von Symbolen einschließlich Beschriftung. Symbole können aus Zeichen bzw. aus bereits definierten Symbolen zusammengesetzt und unter einem dem Anwender gebräuchlichen Namen in der Symboldatei abgelegt werden.
Aus den vorhandenen Symbolen bzw. Grundbildern werden mit Hilfe des Bildeditors die erforderlichen Grafiken interaktiv erstellt. Zum Erleichtern des Editierens stehen dieser Editorebene eine Anzahl von Funktionen zur Verfügung, z. B. zum Löschen, Elementevervielfachen, Verbinden von Elementen, Beschriften, zum Korrigieren, für Blockoperationen, zum Speichern und zur Druckausgabe.

3.2. A3-Editor

Aufbauend auf die mit dem Grundwerkzeug erstellten Zeichengeneratorfiles, Symboldateien und (Teil)-Bilder erlaubt der A3-Editor (als Anwenderprogramm) das Erstellen von Bildern aus Symbolen und Teilbildern in einem wählbaren Format (max. A3 hoch oder quer; etwa 11 KByte).
Der Editiervorgang wird zusätzlich zu den aus GEDIT bekannten Funktionen durch folgende Manipulationsmöglichkeiten unterstützt:

– Auswahl des erforderlichen Formatbereiches, **in** dem gearbeitet wird (Verschieben des im Monitor dargestellten Teilbildes über dem Format),
– Festlegen des Formatbereiches, **mit** dem gearbeitet wird (1..n Zeilen, 1..n Spalten) und
– Funktionsrealisierung mit bzw. im gewählten Bereich verschieben, kopieren, löschen, definiert füllen, von/auf Diskette lesen/schreiben.

Die Autoren arbeiten auf dem Gebiet der Soft- und Hardwareentwicklung für Mikrorechner an der TH Leipzig, Sektion Elektroenergieanlagen.
Im Vordergrund steht die Entwicklung von Grafiksoftware für den Bereich der elektronischen Industrie.
Dieter Herden (43) studierte an der TH Ilmenau und war von 1968 bis 1977 im Kombinat Carl Zeiss JENA tätig. 1971 erfolgte an der TU Dresden im Rahmen einer außerplanmäßigen Aspirantur die Promotion A. Seit 1979 ist Dieter Herden Dozent an der TH Leipzig.
Rolf Lüdicke (29) ist gelernter Elektromonteur. 1983 schloß er sein Studium an der TH Leipzig ab und arbeitet jetzt als Systemprogrammierer.
Claus Wippich (27) ist von Beruf Elektromechaniker. Seit Abschluß seines Studiums an der TH Leipzig im Jahre 1985 ist er als Ingenieur für Lehre und Forschung tätig.

Das bearbeitete Bild kann im Moment der Bearbeitung aus maximal zwei verschiedenen Zeichengeneratoren (Grafik/Standardzeichen) bestehend dargestellt werden. Zum Druck kann es aus vier verschiedenen, ladbaren Zeichengeneratoren bestehen (z. B. für Schriftfelder, Bemaßung). Die Überlagerung der vier verschiedenen Teilbilder wird unterstützt.

3.3. Kommandodateien

Zum schnellen Einarbeiten des Anwenders in das Programmsystem wurde die Möglichkeit des Erstellens und Abarbeitens von Kommandodateien implementiert.
Nach Eröffnen einer Kommandodatei im Hauptmenü werden sämtliche folgende Interaktionen in diese Kommandodatei bis zum Abschluß dieses Modus gespeichert. Zur Demonstration des Grafikeditors bzw. zum Training des Anwenders kann die Kommandodatei aufgerufen werden, so daß in einem wählbaren Zeitregime der Ablauf der Operationen vom Anwender verfolgt werden kann.

4. Schnittstellen

In der vorliegenden Version erzeugt GEDIT folgende Dateien als Schnittstellen zwischen den Programmschichten bzw. zu Anwenderprogrammen:

– Zeichengeneratorfile (128 Zeichen/Datei)
– Symboldatei (Symbole adressierbar über Namen)

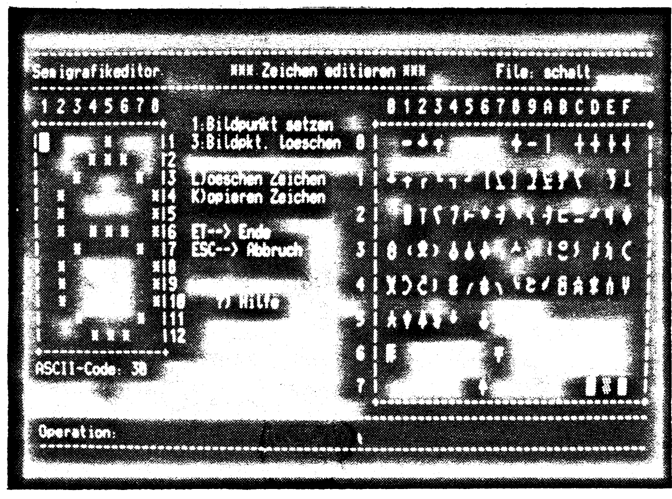


Bild 4 Arbeits- und Menüfeld des Zeicheneditors

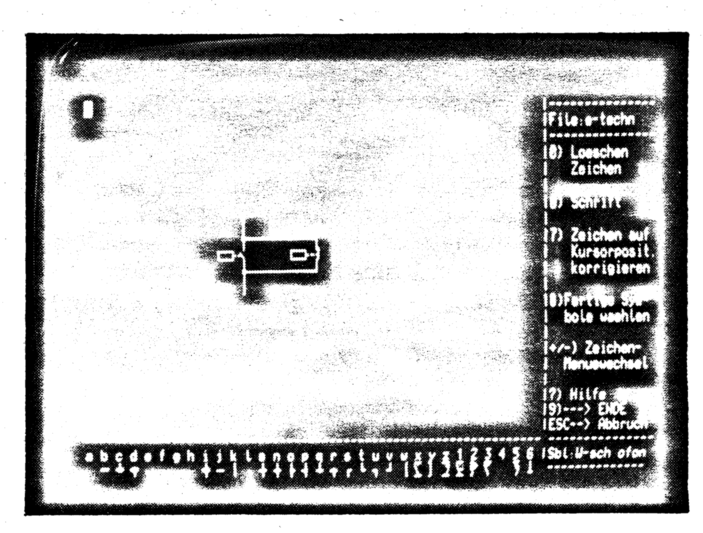


Bild 5 Arbeits- und Menüfeld des Symboleditors

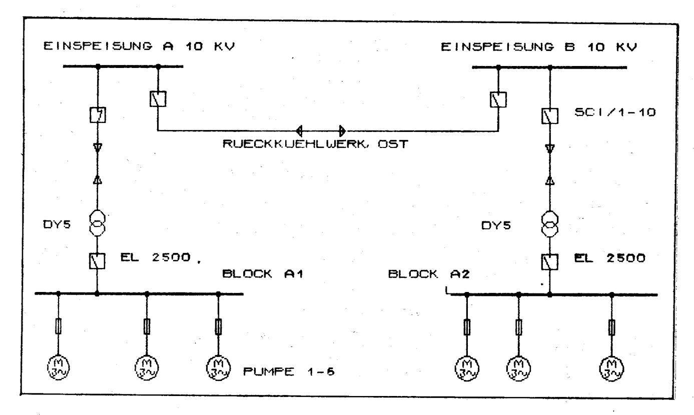


Bild 6 Übersichtsschaltplan

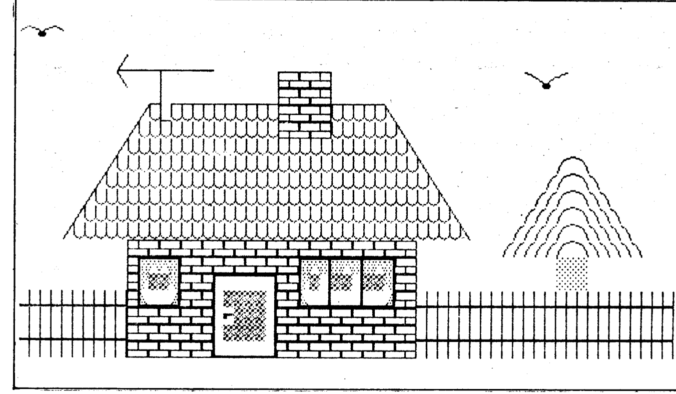
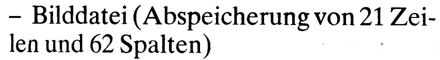


Bild 8 Demonstrationsbeispiel



– Bilddatei (Abspeicherung von 21 Zeilen und 62 Spalten)
– Kommandodatei (z. B. für Analysen, Training, Demonstration).

Der A3-Editor erzeugt über Anwahl noch eine Datei, die Symbolnamen, Positionen und Bezeichnungen enthält. Diese Datei bildet die Voraussetzung zur Generierung einer Strukturmatrix der grafisch eingegebenen Elemente. Das Einbinden von GEDIT-Prozeduren in beliebige Anwenderprogramme ist möglich. Die definierte Programmschnittstelle wird durch Prozeduren wie:

– ZGMOVE: Laden des Zeichengenerators in den parallelen RAM,
– Druck-Prozedur: Ausgabe von Grafiken auf den Drucker und
– Window-Prozeduren zur Symbol- und Bildausgabe und Abspeicherung, zur Bildeditierung, Beschriftung, Textausgabe usw. realisiert.

Die Beschreibung der Anwenderprogrammschnittstelle liegt als Dokumentation und als kommentierte Turbo-Quelle vor.

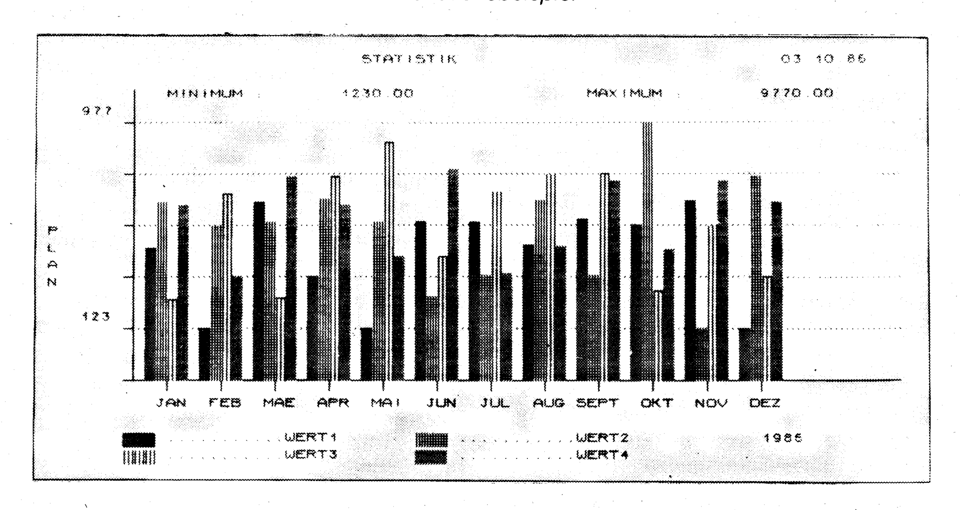


Bild 7 Statistische Darstellung
Fotos (2): V. Pullwitt

5. Charakteristische Anwendungsbereiche

Auf den Bildern 6–8 werden einige auf dem PC 1715 editierte bzw. generierte Grafiken aus unterschiedlichen Anwenderbereichen dargestellt.

Der Grafikeditor ist besonders geeignet für:

① Symbolgrafiken
– Netz-, Trassen-, Stromlaufpläne
– LP-Layouts, Bestückungs- und Anordnungsübersichten
– Piktogramme
– Programmlaufpläne, Steuerungsschemen u. a. m.
② Montageschemen, Ablaufdarstellungen, Werbe- und Demonstrationsgrafiken
③ Erfassen und Verwalten von Norm- und Wiederholteilen mit grafischer Ausgabe sowie die einfache Zeichnungserstellung
④ Statistische Darstellungen (Balken-, Säulendiagramme)

6. Zusammenfassung

Neben branchenorientierter grafischer Anwendersoftware haben allgemeine, nutzerfreundliche Grafikeditoren ihren Platz in der grafischen Informationsverarbeitung auf Personal- und Arbeitsplatzcomputern gefunden /4, 5/. Grafische Editoren haben den Vorteil, daß der Anwender sehr unterschiedliche grafische Aufgabenstellungen interaktiv bearbeiten kann. Darüber hinaus besteht z. T. die Möglichkeit, durch definierte Programm- und Datenschnittstellen den Grafikeditor in Anwenderprogramme einzubinden.
Im gegebenen Fall wird das Leistungsspektrum des PC 1715 dahingehend erweitert, daß der Nutzer für geeignete Anwendungsfälle selbst in die Lage versetzt wird, ohne Programmierkenntnisse sein spezielles grafisches Problem mit Hilfe der angebotenen Softwarewerkzeuge zu bearbeiten. Das Einbeziehen von Grafikprozeduren in Problemprogramme ist dann erforderlich, wenn die grafische Darstellung der berechneten bzw. ausgewählten Elemente und ihrer Strukturbeziehungen die Ergebnisfindung erheblich beschleunigt und eine rationelle Ausgabe der Ergebnisse gewährleistet. Wesentlich ist in diesem Zusammenhang die Möglichkeit, erstellte und manipulierte Grafiken über Prozeduren zur Ermittlung der Art der Bildelemente und ihrer Strukturbeziehungen inhaltlich zu interpretieren.

Literatur

/1/ robotron 1715 – ein neues Erzeugnis der Mikrorechentechnik. NTB **29** (1986) 1
/2/ Wippich, C.: Realisierung einer grafischen Informationsverarbeitung auf Mikrorechnern des K-1520-Systems. Diplomarbeit, TH Leipzig, SEA 1985
/3/ Herden, D.; Lüdicke, R.; Wippich, C.: Nutzerdokumentation GEDIT / V 1.0. TH Leipzig, SEA 1986
/4/ Grabowsky, R.: Computergrafik auf dem Mikrocomputer. Teubner Verlag Stuttgart 1985
/5/ Purgathofer, W.: Ein allgemeiner grafischer Editor. Angewandte Informatik 2/85. Friedr. Vieweg & Sohn Verlagsgesellschaft

✉ KONTAKT ☏

Technische Hochschule Leipzig,
Sektion Elektroenergieanlagen,
Dr. Dieter Herden,
Karl-Liebknecht-Str. 132, Leipzig, 7030

robotron KC 87:

Der neue Kleincomputer im Überblick

Dr. Gert Keller,
Dr. Gunter Kleinmichel
VEB Robtron Meßelektronik
„Otto Schön" Dresden

Seit einiger Zeit findet eine neue Rechner-Kategorie das allgemeine Interesse – der Kleincomputer. Dem ersten Vertreter dieser Kategorie, dem Kleincomputer robotron KC 85/1, folgt nun mit dem KC 87 eine Weiterentwicklung, die dem Anwender durch

- Integration des BASIC-Interpreters in das Grundgerät
- Ergänzung des Modulsortiments um 4 Baugruppen
- Erweiterung des Softwareangebotes

eine Erhöhung des Gebrauchswertes und, das sei nicht verschwiegen, dem Hersteller eine wesentlich rationellere Fertigung garantiert. Da zwischen den Geräten KC 85/1 und KC 87 vollständige Hard- und Softwarekompatibilität bestehen, sind ein problemloses Nachrüsten sowie die Parallelarbeit mit beiden Computertypen möglich.
Das Spektrum der möglichen Einsatzgebiete des KC 87 erstreckt sich von der Rationalisierung von Leitungs- und Verwaltungsaufgaben und der Laborautomatisierung über den Einsatz in allen Zweigen des Bildungswesens bis hin zur Verwendung im individuellen Bereich als Heimcomputer.

Hardware

Da die technischen Eigenschaften des KC 85/1 weitgehend als bekannt vorausgesetzt werden dürften, sollen hier nur noch einmal die Hauptmerkmale des KC 87 erläutert und die technischen Parameter in tabellarischer Form zusammengefaßt werden. Das Grundgerät ist als Kompaktgerät ausgeführt und enthält die Baugruppen Rechnereinheit, Speichereinheit, Tastatur, Bildschirmansteuerung, Peripherieanschlüsse und Stromversorgung. Der Geräte-Bus ist analog dem K-1520-Bus ausgebildet und auf 4 Modulsteckplätze herausgeführt. Die Bildausgabe soll über handelsübliche Fernsehgeräte erfolgen, als Massenspeicher sind Kassettenmagnetbandgeräte vorgesehen. Das durchgängig eingehaltene Prinzp der Modularität bei der Aufrüstung des Grundgerätes ermöglicht eine Erhöhung der Leistungsfähigkeit sowie eine optimale Anpassung an den speziellen Anwendungsfall durch

- Erweiterbarkeit von Arbeits- und Programmspeicher
- funktionelle Erweiterbarkeit, wie z. B. Druckeranschluß, Analog-Digital-Umsetzung, Peripherieerweiterung, EPROM-Programmierung und Spracheingabe.

Dabei wurde darauf geachtet, daß die Ergänzung der Erstausstattung mit Erweiterungsmoduln ohne Eingriffe in den Computer und für jede Serie problemlos möglich sein muß. Beim Bemessen der Stromversorgungseinheit im Grundgerät sind Leistungsreserven für die Versorgung von Moduln und sonstigen Anwenderschaltungen vorgesehen.
In Tafel 1 sind einige Parameter des KC 87 aufgeführt.

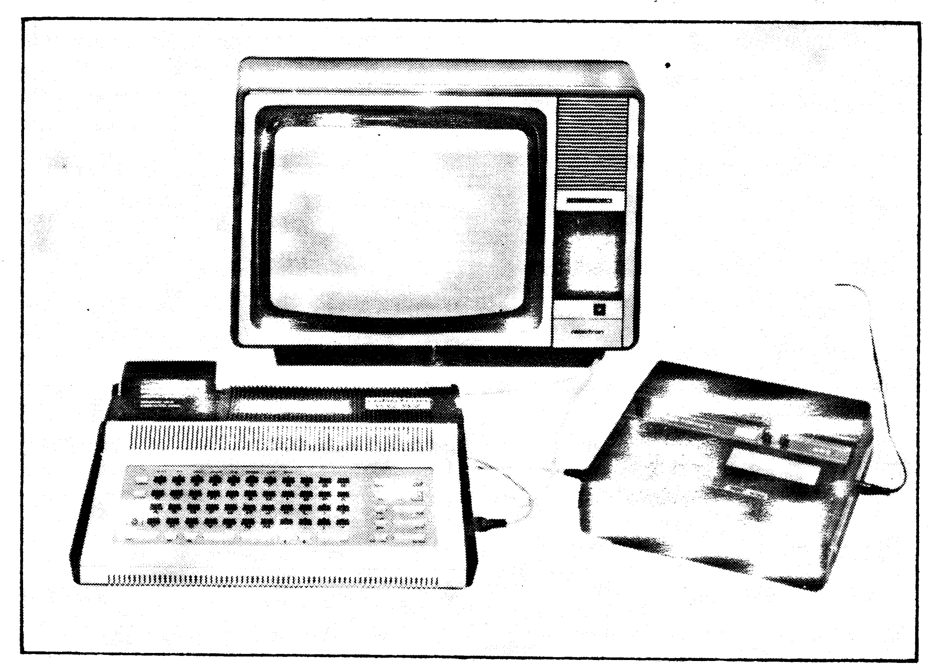


Bild 1 Der KC 87 mit Erweiterungsmodulen, Fernsehgerät und Kassettenmagnetbandgerät

Grundsoftware

Der KC 87 enthält im ROM des Grundgerätes das Betriebssystem (4 KByte) und den BASIC-Interpreter (10 KByte). Zusätzlich steht ein Zeichensatz mit 96 alphanumerischen und 128 Grafikzeichen bereit. Diese Systemsoftware stimmt mit der des KC 85/1 voll überein.
Das Betriebssystem ist in seinem wesentlichen Aufbau am weitverbreiteten CP/M orientiert. Dies betrifft besonders seine Struktur (CCP, BOS, BIOS) sowie die Nutzbarkeit von 32 Unterprogrammen über entsprechende Schnittstellen (CALL 5). Die Unterschiede zum CP/M resultieren aus der Nutzung der Magnetbandkassetten als Massenspeicher.
Der BASIC-Interpreter des KC 87 stimmt in seinem Grundumfang und in

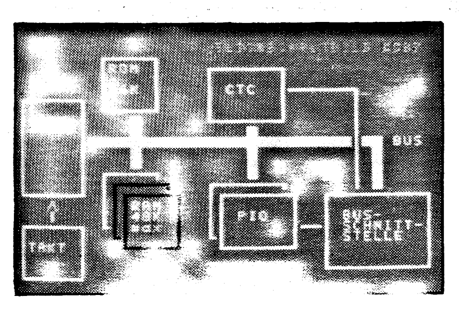


Bild 2 Blockschaltbild des KC 87



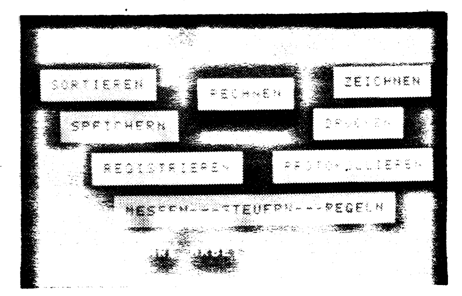
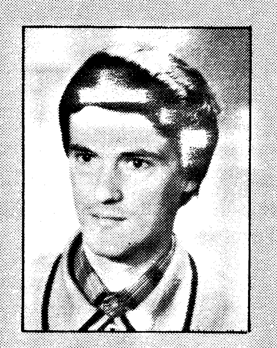


Bild 3 – Bild 6 Der KC 87 ist vielseitig einsetzbar

Gert Keller studierte an der Technischen Universität Dresden Mathematik und absolvierte danach ein Forschungsstudium am Wissenschaftsbereich Numerische Mathematik, das er 1979 mit der Promotion zum Dr. rer. nat. abschloß. Seit 1980 ist Gert Keller im VEB Robotron-Meßelektronik „Otto Schön" tätig. Seine Spezialgebiete sind Numerische Mathematik, Informatik, Computerentwicklung und -anwendung. Seit 1984 ist Gert Keller Softwareverantwortlicher für Kleincomputer. Weiterhin ist er Mitglied des Forschungsrates des VEB Kombinat Robotron.

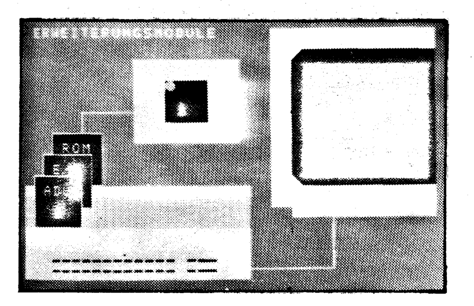


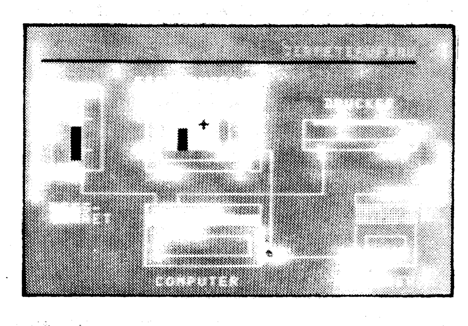
Gunter Kleinmichel studierte an der Technischen Universität Schwachstromtechnik. Nach seinem Studium war er Assistent an der Sektion Informationstechnik der TU Dresden. Dort promovierte er mit einer Arbeit über piezoelektrische Meßgrößenaufnehmer zum Dr.-Ing. Seit 1970 ist Gunter Kleinmichel wissenschaftlicher Mitarbeiter und Abteilungsleiter im VEB Robotron-Meßelektronik „Otto Schön" Dresden. Seine Arbeitsgebiete sind Schwingungstechnik und Akustik, elektronische Meßtechnik, Digitaltechnik und Computerentwicklung.



Tafel 1 Die wichtigsten Parameter des KC 87

Prozessor:	U 880 D
Speicherkapazität:	4 KByte ROM Betriebssystem 10 KByte ROM BASIC-Interpreter 2 KByte ROM Zeichengenerator 16 KByte RAM Anwenderarbeitsspeicher 1 KByte RAM Bildspeicher 1 KByte RAM Farbattributspeicher
Tastatur:	Elastomertastatur mit schreibmaschinenähnlicher Anordnung
Bildausgabe:	Fernsehgerät Anschluß über Antennenbuchse für Schwarzweiß-Darstellung bzw. über RGB-Buchse für Farbdarstellung
Bildaufbau:	24 Zeilen (Standard) 20 Zeilen (wahlweise einschaltbar) 40 Zeichen pro Zeile
Zeichenvorrat:	128 alphanumerische und Steuerzeichen 128 Grafiksymbole für quasigrafische Darstellung
Farbdarstellung:	je 8 Hintergrund- und Vordergrundfarben (nur bei Farbversion KC robotron 87.11)
Tonausgabe:	programmierbar, Ausgabe über eingebauten Summer oder externen Verstärker
Daten-Ein/Ausgabe-Kanäle:	8 Bit, Ein- oder Ausgabe programmierbar je 1 Zähler- und Zeitgeberkanal
Massenspeicher:	Kassettenmagnetbandgerät Übertragungsgeschwindigkeit etwa 1000 Bit/s
weitere externe Anschlüsse:	4 Steckplätze für Erweiterungsmoduln 1 Steuerhebelanschluß
Hauptmaße in mm:	400 × 80/30 × 300
Masse:	3,9 kg
Stromversorgung:	220 V ± 10 %, 25 VA 50 Hz ± 1 Hz
Betriebstemperaturbereich:	+5 °C bis +35 °C





Fotos: Darre (6)

der Syntax weitgehend mit dem international üblichen überein, so daß eine Übernahme von Programmen vergleichbarer Rechner im allgemeinen leichtfällt. Zur Anpassung an die konkrete Hardware wurden im Herstellerbetrieb einige Änderungen und Ergänzungen vorgenommen, die schon beim KC 85/1 wirksam wurden. Zu nennen sind davon die erweiterten EDIT-Möglichkeiten, die in Verbindung mit der angepaßten Tastatur effektive Korrekturmöglichkeiten bieten. Auch die speziellen BASIC-Funktionstasten (RUN, LIST u. a.) erleichtern die Arbeit.
Außerordentlich bewährt haben sich die durch die Anweisungen WINDOW und PRINT AT erweiterten Varianten der Bildausgabe, die die Definition variabler Fenster (Rollbereiche) und die Ausgabe von Informationen an beliebigen Bildschirmpositionen gestatten. Dadurch ist auf einfache Weise eine flexible und übersichtliche Bilddarstellung ebenso möglich wie das schnelle Zeichnen von Bildern oder das Verändern (Bewegen) von Bildteilen. Auch die farbige Gestaltung der Bildschirmausgabe wird durch die speziellen BASIC-Anweisungen INK, PAPER und BORDER unterstützt.
Für viele Anwender, vor allem im Bildungswesen, ist es von Bedeutung, daß der für die Computer KC 85/1 und KC 87 im VEB Robotron-Meßelektronik „Otto Schön" Dresden aufbereitete BASIC-Interpreter auch für die Kleincomputer KC 85/2 und KC 85/3 zur Verfügung gestellt wurde. Damit wurde eine gewisse Kompatibilität beider Computertypen erreicht.

Erweiterungsbaugruppen zum KC 87

Speichererweiterungen

Der verfügbare Arbeitsspeicher für den Anwender, der nach Abzug der vom Betriebssystem und vom BASIC-Interpreter benötigten Arbeitszellen netto 15 086 Bytes beträgt, kann durch RAM-Erweiterungen um jeweils 16 KByte erweitert werden. Bis zu zwei RAM-Erweiterungsmoduln sind steckbar. Durch interne DIL-Schalter wird der Adreßbereich auf 4000H–7FFFH bzw. 8000H–BFFFH eingestellt. Damit sind maximal 47854 Bytes RAM verfügbar.
In Stufen von 10 KByte läßt sich mit dem ROM-Erweiterungsmodul der Programmspeicher aufrüsten. Der Modul enthält 5 Steckfassungen für EPROM-Schaltkreise U 2716 oder äquivalente Typen. Auch hier wird der belegte Adreßbereich intern über DIL-Schalter eingestellt (4000H–67FFH oder 8000H–A7FFH oder C000H–E7FFH).

Programm-Module

Auf der Grundlage der ROM-Erweiterungsmoduln werden Software-Moduln mit folgenden Programmsystemen angeboten:

- SYPS-K-1520-Editor/Assembler
- Interpretativer Dialog-Assembler/Zusatzmonitor.

Gegenüber den seit längerer Zeit bereits auf Magnetbandkassetten im Vertrieb befindlichen RAM-Versionen dieser Programmsysteme erlauben die Programm-Moduln eine rationellere und komfortablere Arbeitsweise. Beide Moduln belegen den Adreßbereich C000H–E7FFH, sind also nicht gleichzeitig mit dem im Grundgerät auf dem gleichen Adreßbereich installierten BASIC-Interpreter betreibbar.
Beim Einstecken dieser Module in den KC 87 wird der BASIC-Interpreter automatisch abgeschaltet.

Funktionelle Erweiterungsbaugruppen

Das bereits seit 1985 in Vertrieb befindliche Sortiment wurde weiter ergänzt und enthält jetzt Moduln für folgende Zusatzfunktionen:

● *Anschluß von Druckern und elektronischen Schreibmaschinen mit V24-Interface*

Das Druckertreiberprogramm ist im Modul auf einem ROM installiert und belegt den Adreßbereich B800H–BFFFH. Dieser ROM ist mittels DIL-Schalter abschaltbar. Damit kann der gesamte 16 KByte-Bereich ab Adresse 8000H durch einen RAM-Erweiterungsmodul genutzt werden. Da Drucker und elektronische Schreibmaschinen mit unterschiedlichen Interface-Steckverbindern vertrieben werden, sind gegenwärtig drei unterschiedliche Typen von Anschlußmoduln im Angebot.

● *Analog-Digital-Umsetzung*

Dieser Modul ermöglicht die gleichzeitige Messung von 4 Gleichspannungen im Bereich −99 mV ... +999 mV bei einer Geschwindigkeit von maximal 168 Umsetzungen/s.
Für die Analog-Digital-Umsetzung wird der Schaltkreis C 520 D eingesetzt. Der Modul kann unter zwei unterschiedlichen Portadressen betrieben werden.

● *Erweiterung der digitalen Ein-/Ausgabe-Kanäle*

Neben den im Grundgerät am Anwendersteckverbinder verfügbaren 8 digitalen E/A-Kanälen sind in jedem E/A-Modul weitere 16 freiprogrammierbare Kanäle verfügbar. Über DIL-Schalter sind zwei unterschiedliche Portadressen einstellbar.

● *Programmierung von 2-KByte-EPROM (U 2716)*

Der Modul ermöglicht das Lesen, Prüfen und Programmieren von 2-KByte-EPROM in den bzw. aus dem Speicher des Kleincomputers. Für die Aufnahme der Speicherschaltkreise ist eine Schwenkhebelfassung auf dem Modul angebracht.

● *Spracheingabe*

Ein Wortschatz von 50 Worten wird durch den Spracheingabemodul erkannt und die Nummer des identifizierten Wortmusters in einer Arbeitszelle zur Weiterverarbeitung bereitgestellt. Der Wortschatz muß in einer „Lernphase" vorher dem Spracherkennungsprogramm vermittelt werden. Die Spracheingabe erfolgt über ein mitgeliefertes Kopfbügelmikrofon.

Alle aufgeführten Erweiterungsbaugruppen sind in einem einheitlichen Modulgehäuse untergebracht, das wahlweise auf einen der vier Modulsteckplätze des KC 87 eingeschoben werden kann.

Darüber hinaus gehören zum Kleincomputersystem KC 87 noch folgende Baugruppen:

● Farbmodul zur Umrüstung der Schwarz/weiß-Variante KC 87.10 auf Farbwiedergabe über RGB-Anschluß von Farbfernsehgeräten (Dieser Modul wird auf einen speziellen Steckverbinder im Computer gesteckt).
● Ergänzungssatz Farbe zur Nachrüstung eines RGB-Einganges an Farbfernsehgeräten
● Steuerhebel
● Steuerhebeladapter zum Anschluß von 2 Spielhebeln
● Zugentlastung für Moduln mit seitlichem Kabelausgang
● Adapter für den Übergang von 15poliger Buchse auf eine Klemmleiste.

Softwareangebot

Zum Lieferumfang des KC 87 gehört eine Programmkassette mit Demonstrationsprogrammen, die den Rechner und seine Anwendungsmöglichkeiten erläutern. Die außerdem für den KC 87 auf Magnetbandkassetten bzw. ROM-Moduln angebotene Software umfaßt sowohl sogenannte Systemprogramme (z. B. zur Assemblerprogrammierung) als auch zahlreiche BASIC-Anwenderprogramme aus den Bereichen Wissenschaft und Technik, Lehre und Lernen, Datenverarbeitung sowie Spiele.
Zur Unterstützung der Programmierung in Assembler werden ein Editor/Assembler (ASM) und ein interpretativer Dialogassembler (IDAS) angeboten. Für denjenigen, der auf Maschinencodeebene arbeiten muß, steht ein Zusatzmonitor (ZM) zur Verfügung. Diese drei Programme werden auch als ROM-Version auf entsprechenden Moduln angeboten.
BASIC-Anwenderprogramme stehen auf bisher 10 Programmkassetten bereit. Diese Programme sollen die vielseitigen Einsatzmöglichkeiten des Computers deutlich werden lassen.
Auf die Gestaltung dieser Programme wurde besonderer Wert gelegt, so daß der Nutzer bezüglich Bildschirmgestaltung und Dialogführung Anregungen erhält. Einige Programme, unter anderem solche zur numerischen Mathematik und Statistik, dienen der effektiven Lösung konkreter praktischer Aufgabenstellungen.
Das Softwareangebot wird ergänzt durch das Textverarbeitungsprogramm TEXT 1 und in Kürze durch die Programmiersprachen PASCAL und FORTH.

Einige Anwendungsgebiete

Die wichtigsten Einsatzgebiete eines Computers resultieren in der Regel aus seinen Haupteigenschaften. Neben dem geringen Preis, seiner unkomplizierten und einfachen Bedienbarkeit sind das beim KC 87 vor allem:

- Schnelle und stabile Bildausgabe
- gute Eignung der Quasigrafik für flächenhafte und bewegte Bilder
- einfache Modifizierbarkeit für verschiedene Aufgaben durch Steckmodule
- flexible analoge und digitale E/A-Möglichkeiten.

Mit diesen Eigenschaften ist der Computer gut zur Heranführung praktisch aller Bevölkerungsgruppen an Probleme der Anwendung der Computertechnik geeignet.
Neben diesen Ausbildungszielen auf dem Gebiet der Informatik eignet er sich aber auch zur Wissensvermittlung und zum Wissenstest in anderen Fächern, wie z. B. Biologie, Geografie und Geschichte.
Ebenso zeigen die an der Pädagogischen Hochschule Güstrow durchgeführten Arbeiten überzeugend die Eignung des robotron-Kleincomputers zur Vermittlung von Grundlagen der Automatisierung im Rahmen des obligatorischen und fakultativen polytechnischen Unterrichts. So können durch Kopplung der Schülerexperimentiergeräte mit dem KC 87 viele interessante Steuer- und Regelaufgaben erläutert und ausgeführt werden. Im industriellen Einsatz hat sich der Rechner besonders bei der Laborautomatisierung bewährt. Hier fallen seine Erweiterungsmöglichkeiten zum Anschluß elektronischer Geräte und Versuchseinrichtungen positiv ins

Gewicht. Durch die (bei vergleichbaren Computern selten vorhandenen) vielfältigen Anschlußmöglichkeiten zur Übertragung digitaler und analoger Signale wurden in einigen Anwendungsfällen bereits erhebliche Effekte erzielt (Zeiteinsparung, umfassende Auswertung aller Meßdaten).
Wegen der relativ hohen Rechen- und Bildausgabegeschwindigkeit lassen sich auch gewisse wissenschaftlich-technische Berechnungen effektiv ausführen. Auch kleinere Verwaltungsaufgaben (z. B. für Struktureinheiten bis etwa 50 Mitarbeiter) sind lösbar, wobei durch die Begrenzungen des Speicherplatzes und der Rechengeschwindigkeit natürlich Grenzen gesetzt sind. Insgesamt zeigt sich, daß es viele Aufgaben gibt, für die der Kleincomputer KC 87 effektiv einsetzbar ist.

Vertriebsweise

Der Vertrieb des Kleincomputers robotron KC 87 erfolgt durch territoriale Vertriebsbetriebe des VEB Kombinat Robotron. Die angebotene Software wird durch den VEB Robotron-Vertrieb Berlin, Abt. VD, PF 1235, Berlin, 1086 vertrieben. Dort ist auch die folgende Literatur bestellbar: Kleincomputer robotron KC 87, Bedienungsanleitung; Kleincomputer robotron KC 87, Programmierhandbuch; Betriebssystem robotron KC 85/1 und KC 87; Softwareangebot zum Kleincomputer KC 85/1.

Literatur

/1/ Keller, G.: Systemsoftware für die Kleincomputer KC 85/1 und KC 87. NTB **30** (1986) 6

Der Kleincomputer als Prüfbildgenerator

Dr. Bernhard Müller
VEB Mikroelektronik „Wilhelm Pieck“ Mühlhausen
Heinz-Joachim Peist
Pädagogische Hochschule „Dr. Theodor Neubauer“ Erfurt/Mühlhausen

Im folgenden wird ein Programm vorgestellt, das jeden KC 85/2 bzw. KC 85/3 zum Prüfbildgenerator macht. Die Autoren verfolgen mit dem Videoprüfprogramm das Ziel, die Farbtüchtigkeit und Grafikfähigkeit der Kleincomputer KC 85/2 und KC 85/3 aus Mühlhausen in Verbindung mit deren TV-Anschlußmöglichkeiten (Antennen-, Video- und RGB-Ausgang) für Testzwecke in der Fernsehtechnik zu nutzen. Dieses Programm ist gleichzeitig ein Beispiel dafür, wie sich teure Hardware durch Software ersetzen läßt. In Fernsehzentralen sowie in den Service-Werkstätten der Dienstleistungsbetriebe des Rundfunk- und Fernsehhandwerks werden Prüf- bzw. Testbilder zur Fehlerdiagnose an Fernsehgeräten bzw. deren Baugruppen benötigt. Im allgemeinen werden dazu Testbildgeneratoren verwendet, die meist sehr kostenaufwendig sind. Mit einem KC 85/2 oder KC 85/3 können nun die für solche Prüfarbeiten notwendigen Testbilder per Software bereitgestellt werden. Die Prüfbilder des Videoprüfprogramms wurden auf die im Service üblichen Arbeiten bzw. Kontrollen abgestimmt.

Im einzelnen sind dies:

Rotfläche (Bild 2; die Farbbilder sind auf der 4. Umschlagseite angeordnet)
- Kontrolle der Farbreinheit
- Kontrolle des Weißabgleichs

Kreis (Bild 1)
- Kontrolle und Justage von Geometriefehlern

Schachbrett (Bild 4)
- Kontrolle und Justage von Geometriefehlern
- Kontrolle des Farbdecoders
- Kontrolle des Weißabgleichs

Konvergenzgitter (Bild 5)
- Erkennen von Konvergenzfehlern
- Erkennen von Trapezfehlern
- Bildlagejustierung
- Kontrolle der Zeilensynchronisation bei Videorecordern

Farbbalken (Normalfarbbalken) (Bild 6)
- Kontrolle der Farbwiedergabe
- Justage des Decoders
- Einstellung der Farbendstufen
- bei schwarz-weiß-Geräten (Grautreppe); Kontrolle der Videoendstufe

Für die möglichen Einstellungen und Justagen hält man sich an die in den Serviceunterlagen vom Fernsehgerätehersteller gegebenen Servicehinweise.
Jeder Kleincomputer aus Mühlhausen verfügt neben dem Antennenausgang auch über einen Videoausgang (1 V Spitze-Spitze an 75 Ω) und über einen RGB-Ausgang (ebenfalls 1 V Spitze-Spitze an 75 Ω). Übliche Farbservicegeneratoren verfügen in der Regel über keinen RGB-Ausgang. Bei weiterer Verbreitung dieser Einspeisungsart an Farbgeräten wird somit der KC zu einem guten Helfer bei Prüfarbeiten. Den Vorteil der RGB-Einspeisung gegenüber der HF-Einspeisung verdeutlichen die Bilder 7 und 8. In beiden Abbildungen ist der gleiche Ausschnitt des Farbbalken-Prüfbildes (Bild 6) dargestellt, in Bild 7 bei HF-Einspeisung vom Antennenausgang des Computers und in Bild 8 bei RGB-Einspeisung.
Der Videoausgang ermöglicht es, den Computer zur Einspeisung an Videorecordern zu verwenden. Die Prüfsignale, besonders das des Konvergenzgitters sowie das des Farbbalkens, lassen sich bei der Herstellung von Videoprogrammen als Kontrollvorspann auf Videobändern verwenden. Das Menübild des Videoprüfprogrammes (Bild 3) wurde aus optischen Gründen recht aufwendig gestaltet. Es enthält alle Vordergrund- und alle Hintergrundfarben, die mit dem Computer erzeugt werden können. In Anlehnung an die Lage der Farbvektoren in der Chrominanzebene des PAL-Systems wurden die Vordergrundfarben im Farbkreis angeordnet. Das Menübild gibt außerdem einen kleinen Einblick in die Möglichkeiten, die dem Nutzer dadurch gegeben sind, daß er eigene Grafikzeichen im Programm verwenden kann (siehe Umrandung des Farbkreises und Unterschriften). Dieses Bild läßt sich z. B. als Pausen- oder Stationsbild in Fernsehzentralen verwenden.
Das Videoprüfprogramm ist in BASIC geschrieben und wird durch einige Maschinenprogramme unterstützt. Die Maschinenprogramme werden zusammen mit dem BASIC-Programm geladen.
Das Speicherangebot der Grundgeräte KC 85/2 und KC 85/3 ist für das Videoprüfprogramm ausreichend.
Das beschriebene Programm wurde auf der RFT-Fachtagung der Bezirke Erfurt, Gera und Suhl vorgestellt, die am 5. und 6. November 1985 in Oberhof stattfand. Es wurde dort mit guter Resonanz von den Fachkollegen, besonders auch von den Service-Technikern aufgenommen.
Interessenten können über das Neuererbüro der Pädagogischen Hochschule das Videoprüfprogramm beziehen.

Fotos: Peist (8)

⊠ KONTAKT ☏

Pädagogische Hochschule „Dr. Theodor Neubauer“ Erfurt/Mühlhausen, Hochschulbereich Erfurt, Neuererbüro, Koll. Dargel, Nordhäuserstraße 63, Erfurt, 5064

Programmpaket für Projektierung

An der Ingenieurschule für Maschinenbau und Elektrotechnik Berlin, speziell in der Abt. Elektroenergieanlagen, wurde für das Fachgebiet „Projektieren elektrotechnischer Anlagen" ein Programmpaket mit folgenden Programmen erstellt:

● Beleuchtungsberechnung
– Wirkungsgradmethode für Innenraumbeleuchtung
– Wirkungsgradmethode für Außenbeleuchtung
– Technisch-ökonomischer Variantenvergleich
– Lichtstärkemethode für linienförmige und rotationssymmetrische Leuchten
● Netzberechnungen
– vermaschte Netze
– Kurzschlußstromberechnung
– Nullungsnachweis
– Strom- und Spannungsverteilung in Ringnetzen
– Motoranlauf.

Diese Programme sind einsatzfähig auf den Rechnertypen PC 1715, KC 85/1 und KC 85/2, K 1000.

Ingenieurschule für Maschinenbau und Elektrotechnik Berlin, Abt. Elektroenergieanlagen, Marktstr. 9, Berlin, 1134, Tel. 5 50 93 01. Dr. Bätz

PC 1715 jetzt mit Betriebssystem UDOS

Im ZFT des KEAW wurde das Betriebssystem UDOS auf dem PC 1715 implementiert (Geräteversion mit Bildschirm 24 × 80 und Minifolienspeicher MFS 1.6). Als Dateiverwaltungssystem wurde NDOS verwendet. Beim Standard-Diskettenformat mit 80 Spuren × 32 Sektoren × 256 Byte pro Sektor (DS–DD) sind 640 KByte pro Diskette speicherbar. Weiterhin sind die Formate 80 × 16 × 256 (SS–DD) und 40 × 16 × 256 (SS–SD) zugelassen. Mit Hilfe eines nachladbaren Treibers sind außerdem die Disketten mit dem speziellen UDOS-Format des Bürocomputers A 5120 (40 × 26 × 128) kopierbar, womit ein unproblematischer Übergang vom A 5120 zum PC 1715 gesichert wird.

VEB KEAW „Friedrich Ebert", Zentrum für Forschung und Technologie, Abt. WAC, Storkower Straße 101, Berlin, 1055, Tel. 4 38 84 84. Raehse

Feinjustagevorrichtung für Floppy MKF 3200

Die Floppy-Laufwerke MKF 3200 sind üblicherweise für den Softsector-Betrieb ausgelegt. Um die Anforderungen der Indexloch-Erkennung für den Hardsector-Betrieb realisieren zu können, wurde eine mechanische Feinjustagevorrichtung entwickelt. Mit dieser ist eine schnelle und effektive Lösung zur Justage der Indexloch-Erkennung geschaffen worden. Es wird der Vorteil der wesentlich schnelleren Zugriffzeiten bei Hardsector-Betrieb ohne wesentliche Umbauten am MKF 3200 nutzbar; der Softsector-Betrieb ist auch weiterhin uneingeschränkt möglich.

VEB Schiffselektronik Rostock, Postschließfach 85, Rostock, 2910, Tel. 81 20. Bull/Jähn

LC-80-Assemblerliste

In unserem Rationalisierungs- und Konstruktionsbüro wurde eine ausführlich kommentierte Assemblerliste des Monitorprogramms des LC 80 erarbeitet. Die Liste kann im MEOS-Format (SYPS 1520) oder UDOS-Format (für Bürocomputer) bestellt werden. Sie hat einen Umfang von 24 Seiten. Interessenten können Quelltext und Objektcode auch auf kundeneigenem Datenträger (MB-Kassette oder 8-Zoll-Diskette) beziehen. Die Nachnutzungsgebühr beträgt 60,– M.

Ingenieurschule für Lebensmittelindustrie, Bahnhofstraße 7, Gerwisch, 3101, Tel. Biederitz 308. Dr. Böhlke

---



Den nachfolgenden Beitrag „Die Arbeit der GIDDR in den Jahren 1986–1990" von Prof. Dr. Dieter Hammer, Vorsitzender der Gesellschaft für Informatik der DDR, haben wir in etwas gekürzter Form aus dem Mitteilungsheft Nr. 2/86 der GIDDR entnommen.

Die Arbeit der GIDDR in den Jahren 1986–1990

Die Schwerpunkte der Tätigkeit der GIDDR leiten sich aus den volkswirtschaftlichen Zielen entsprechend den Beschlüssen des XI. Parteitages der SED ab. In der Direktive zum Fünfjahrplan 1986–90 wird zum Ausdruck gebracht, daß die Entwicklung der Volkswirtschaft der DDR in den nächsten Jahren eine hohe Dynamik des Wachstums der Produktivität und Effektivität in allen Bereichen des volkswirtschaftlichen Reproduktionsprozesses erfordert. Zur Sicherung der dynamischen und effektiven Entwicklung der Produktivkräfte ist dazu die rasche Entwicklung und umfassende Anwendung der Schlüsseltechnologien, insbesondere der Mikroelektronik, der Rechentechnik, der rechnergestützten Projektierung, Konstruktion, Produktionsvorbereitung und -steuerung (CAD/CAM) sowie der flexiblen automatischen Fertigungssysteme konsequent weiterzuführen. Diese neue Etappe der intensiv erweiterten Reproduktion der Volkswirtschaft erfordert die enge Verbindung des wissenschaftlich-technischen Neuerungsprozesses mit der breiten Anwendung der wissenschaftlichen Arbeitsorganisation, um neueste Technik mit hochproduktiven Technologien und die Automatisierung ganzer technologischer Prozesse unter Anwendung der Mikroelektronik beschleunigt einzuführen. Die Sicherung des erforderlichen Bildungsvorlaufs und die Weiterbildung der Werktätigen haben dabei eine erstrangige Bedeutung.
Die gesamte Arbeit der GIDDR wird sich entsprechend dem Charakter einer wissenschaftlichen Gesellschaft in diese Prozesse einordnen.

Dabei geht es

– um die Unterstützung der Vorlaufforschung und der Anwendung von Ergebnissen zu den Schwerpunkten und Hauptrichtungen,
– um die Förderung des Erreichens und des Ausbaus von Spitzenpositionen auf Teilgebieten,
– um die Förderung einer bewußten Beobachtungsforschung zur Auslotung der Breite des Wissenschaftsgebietes für das Erkunden neuartiger Lösungen,
– um eine zunehmend bessere Beherrschung komplexer Vorgänge durch die Anwendung dieser Informatik.

Für die Orientierung der Arbeit der GIDDR sind die genannten Ziele unter Berücksichtigung des Komplexprogramms des RGW mit Zeithorizont Jahr 2000 zu sehen. Die Informatik und ihre Anwendung haben einen außerordentlich großen Einfluß auf die Entwicklung der Volkswirtschaft bzw. auf das gesamte gesellschaftliche Leben. Damit verbunden ergeben sich Fragen zu gesellschaftlichen, ökonomischen und sozialen Aspekten, deren Beratung und Diskussion im Rahmen einer neu zu bildenden Fachsektion in der GIDDR erfolgen wird. Neben den speziellen Aufgaben in den Fachsektionen wird sich die Gesamtarbeit der GIDDR in den kommenden Jahren jedes Jahr auf einen besonderen Schwerpunkt konzentrieren:
1987 – Möglichkeiten zur effektiven Entwicklung von Software
1988 – Der massenhafte Einsatz von Computertechnik
1989 – Die breite Einführung und Anwendung von modernen Kommunikationstechniken
1990 – Computerarchitekturen zur Lösung von Aufgaben neuer Anwendungsbereiche
Diese Schwerpunktsetzung beinhaltet die Durchführung von Veranstaltungen und Tagungen der GIDDR insgesamt, in denen auch aus der Sicht aller Fachsektionen eine Auseinandersetzung mit der jeweiligen Thematik erfolgt. Entsprechend dem Statut der Gesellschaft erfüllt die GIDDR ihre Aufgaben zur Entfaltung des wissenschaftlichen Lebens auf dem Gebiet der Informatik durch wissenschaftliche Tagungen, Beratungen und Kolloquien, durch die Unterstützung der wissenschaftlichen Publikationstätigkeit, vor allem durch die Herausgabe ihres Mitteilungsblattes sowie durch Empfehlungen zur Gestaltung der Aus-, Weiter- und Schulbildung auf dem Gebiet der Informatik.
Ihre Informationstätigkeit dient auf der einen Seite der Weiterbildung ihrer Mitglieder und der Vermittlung neuester wissenschaftlicher Erkenntnisse auf dem Gebiet der Informatik, auf der anderen Seite der Kommunikation zu fachspezifischen Fragen auf Spezialkonferenzen und -kolloquien.

KDT-Fachtagung „Lokale Netze – Stand und Entwicklungen"

Der Fachausschuß 7 *Rechnersysteme und -anwendungen* in der Wissenschaftlich-technischen Gesellschaft für Meß- und Automatisierungstechnik (WGMA) veranstaltete am 14. 5. 1986 gemeinsam mit der Universitätsorganisation der KDT der Technischen Universität Dresden und mit bewährter organisatorischer Unterstützung des KDT-Bezirksvorstandes Dresden die Fachtagung „Lokale Netze – Stand und Entwicklungen". Auf der unter Leitung des Fachunterausschusses 7.1 (Prof. Dr. H. Löffler, TUD, Sektion Informationsverarbeitung) durchgeführten Veranstaltung wurde erstmals dem Informationsbedürfnis eines breiten Interessentenkreises in der DDR zur Darstellung dieser Problematik über den bisherigen Rahmen spezifischer Fachgremien hinaus entsprochen.
An der Fachtagung nahmen etwa 500 Vertreter aus dem Bereich des Hoch- und Fachschulwesens, der Akademie der Wissenschaften der DDR, der Industrie, dem Bauwesen und weiteren staatlichen Einrichtungen teil.
Nach einem Überblick von *Prof. Dr. Löffler* über den internationalen Stand und sich abzeichnende Trends bei dem Entwurf und dem Einsatz von LAN sowie einer Bestandsaufnahme der im Bereich des MHF und der AdW der DDR sowie im Kombinat Robotron vorliegenden Ergebnisse und Erfahrungen wurden repräsentative Anwendungslösungen für die Automatisierung, rechenzentrumstypische Einsätze und Büroautomatisierung aus system- und anwendungstechnischer Sicht in mehreren Vorträgen vorgestellt:
Dr. Hackler referierte über das in der Automatisierung diskreter oder kontinuierlicher Prozesse einsetzbare lokale Rechnernetz LOTUNET 8.3 der TU Dresden. Entsprechend den Anforderungen einer Echtzeitinformationsverarbeitung wird als Mediumzugriffsverfahren die deterministische Token-Bus-Zugriffsmethode verwendet. Bei Einsatz des seriellen Ein-/Ausgabeschaltkreises SIO U 856 in der Busansteuerung ist eine max. Datenübertragungsrate von 500 KBit/s festgelegt. Als Ausblick wurde auf das derzeit in Entwicklung befindliche LOTUNET 8.4 eingegangen, mit dem wesentliche Charakteristika des *MAP (Manufacturing Automation Protocol)-Projektes* berücksichtigt werden.
Prof. Garbe berichtete am Beispiel des seit November 1985 an der Ingenieurhochschule Dresden in Erprobung/Einsatz befindlichen IHDnet (Zugriffsverfahren nach dem Polling-Prinzip) über die LAN-Spezifika für rechenzentrumstypische Anwendungen. Beim IHDnet sind die Endsystemrechner ES 1022, SM 4–10 und A 5120 über Knotenrechner K 1520 an einen externen Bus (Koaxialkabel, max. Datenübertragungsrate 500 KBit/s) angeschlossen. *Prof. Zaremba* stellte das Lokalnetz LANCELOT vor, das mit einem eigenen kollisionsfreien Zugriffsverfahren und Lichtwellenleiter-Medium arbeitet und im Rahmen eines Experimentalsystems (A 5601, A 5120 und K 1520) erfolgreich erprobt wurde. Weiterführende Entwicklungsarbeiten sind als LANCELOT 2 mit einer Datenübertragungsrate von 10 MBit/s in Bearbeitung. In den zwei Beiträgen des VEB Kombinat Robotron wurde von *Zumpe* und *Dr. Richter* die auf der Leipziger Frühjahrsmesse 1986 gezeigte experimentelle LAN-Lösung ROLANET 1 vorgestellt. Die Lokalnetzfähigkeit der einzelnen Computerarbeitsplätze wird hardwaremäßig über spezielle, im jeweiligen Hostrechner integrierbare LAN-Controller (Netzwerkinterfaceeinheiten) LNC1–1520, LNC1–1715 realisiert. Der Controller sendet und empfängt mit einer Übertragungsrate von 500 KBit/s. Als Transceiver (Mediumanschlußeinheit), der bis 50 m vom Hostrechner entfernt installiert werden kann, wird ein für alle Rechner einheitlicher Typ eingesetzt. Für die Zuteilung des von allen Arbeitsstationen gemeinsam benutzten Übertragungskanals wurde das CSMA/CD-Verfahren realisiert. Als Übertragungsmedium wird Koaxialkabel (max. 1 km) verwendet. Für die Implementation der LAN-Software kommt für die Büro- und Personalcomputer als Basisbetriebssystem SCP zum Einsatz.
Dr. Gütter erläuterte grundlegende Charakteristika und Einsatzmöglichkeiten moderner Kommunikationsdienste sowie einige der damit verbundenen Anforderungen an die Hard- und Software. Davon ausgehend wurden exemplarisch im Rahmen einer kleinen Systemkonfiguration (drei Bildschirmgeräte robotron K 8912) von einem Kollektiv unter Leitung von Prof. Löffler Anwendungsmöglichkeiten auf der Basis des Mailbox-Dienstes vorgeführt.
Die Veranstaltung kann als ein gelungener erster Beitrag der WGMA/KDT zur breiten Vermittlung von Informationen und Erfahrungen bei dem Entwurf, der Realisierung und dem Einsatz von lokalen Netzen gewertet werden.

Dr. Kuntsche/Prof. Dr. Löffler

2. Symposium zum Einsatz sowjetischer SKR-Technik in der DDR

Das 2. Symposium zum Einsatz sowjetischer SKR-Technik wurde vom 4.–5. 2. 1986 an der Ingenieurhochschule Dresden mit 254 Teilnehmern aus der DDR und 14 sowjetischen Gästen durchgeführt. Träger des Symposiums waren, neben der Ingenieurhochschule Dresden als Veranstalter, die Kooperationsgemeinschaft SM 3/SM 4, der VEB Robotron-Anlagenbau Leipzig und das sowjetische Außenhandelsunternehmen Elektronorgtechnika Moskau.
Die Zielstellung des Symposiums bestand in der Vorstellung der neuen sowjetischen Kleinrechenanlage SM 14–20 und sowjetischer Softwarepakete sowie in der Vermittlung wertvoller Anwendungserfahrungen von DDR-Anwendern und der Förderung des Erfahrungsaustausches zwischen den Anwendern in der DDR. Das Vortragsprogramm umfaßte insgesamt 29 Vorträge, davon 9 Vorträge von den sowjetischen Gästen und 17 Vorträge von Anwendern der sowjetischen SKR-Technik.
In der Plenarsitzung wurden fünf Vorträge gehalten: Von *Prof. Dr. Tzschoppe*, Rektor der Ingenieurhochschule und Vorsitzender des Beirates Informatik beim Ministerium für Hoch- und Fachschulwesen, wurde in seinem Vortrag „Stand und Perspektiven des Einsatzes der SKR-Technik in der Ausbildung auf dem Gebiet der Informatik" die wachsende Bedeutung der SKR-Technik für die Volkswirtschaft der DDR unterstrichen, die sich auch in der Ausbildung widerspiegelt, sowie die neue Ausbildungskonzeption auf dem Gebiet der Informatik in der DDR vorgestellt. *Dr. Platonow* (Ministerium für Gerätebau und Automatisierungssysteme Moskau) stellte Neuentwicklungen der SKR-Technik und peripherer Geräte in der UdSSR vor und *Dr. Garusow* (NPO Zentrprogrammsistem Kalinin) gab eine Übersicht über problemorientierte Anwenderprogrammkomplexe für die SKR-Technik. *Gen. Ulrich* (Betriebsdirektor des VEB Robotron-Anlagenbau Leipzig) zeigte die Perspektiven des Imports von SKR-Technik im neuen Planungszeitraum auf. *Dr. Horn* (Vorsitzender des Problemrates der Kooperationsgemeinschaft) gab einen Überblick über Aufgaben und Ergebnisse der Zusammenarbeit der Anwender in der Kooperationsgemeinschaft SM 3/SM 4 in den fünf Jahren ihres Bestehens.
Das Kleinrechnersystem SM 14–20 wurde von Ljamez (ELORG Berlin) vorgestellt, wobei im Vortrag speziell auf Installations- und Wartungsprobleme eingegangen wurde. Außerdem war die Besichtigung eines der ersten in der DDR installierten Kleinrechnersysteme SM 14–20 im Zentralinstitut für Festkörperphysik und Werkstoffkunde der AdW im Rahmen des Tagungsprogramms vorgesehen.
In einem Rundtischgespräch wurden Fragen der weiteren Verbesserung der Zusammenarbeit der Anwender mit dem Importeur und Exporteur und Fragen der Entwicklung diskutiert.
Ein Sonderheft mit den auf dem Symposium gehaltenen Vorträgen steht zur Verfügung.

Dr. Thomas Horn

CAD/CAM in der ČSSR

28. Internationale Maschinenbaumesse Brno

In diesem Planjahrfünft soll in unserem Nachbarland ČSSR die Produktion in der Elektroindustrie im Vergleich zur vorangegangenen Planperiode um 60 bis 65 Prozent, das heißt im Jahresdurchschnitt um etwa 12 Prozent, erhöht werden. Eine besondere Steigerung erfährt dabei die Produktion elektronischer Bauelemente (um 136 Prozent), von Meßtechnik (um 110 Prozent) und von Rechen- und Automatisierungstechnik. Der Elektronikeinsatz insbesondere im Maschinenbau ist eine Aufgabe, die in der ČSSR mit allem Nachdruck gestellt wird. Deswegen ist es nur folgerichtig, daß die 28. Internationale Maschinenmesse Brno unter dem Leitmotiv „Elektronikeinsatz im Maschinenbau" stand. Vom 17. bis 24. September 1986 zeigten rund 2500 Aussteller aus 30 Ländern in der mährischen Metropole auf einer Gesamtfläche von 115 000 m² u. a. mikroelektronisch gesteuerte Maschinen, komplexe Fertigungszellen sowie Erzeugnisse der Elektrotechnik. Die Messe brachte auch zum Ausdruck, daß die ČSSR dem breiten Einsatz von CAD/CAM-Systemen in der Volkswirtschaft eine hohe Bedeutung beimißt. (In der ČSSR wird für CAD/CAM häufig der Begriff „Automatisierung der Ingenieurarbeiten" verwendet.) Sowohl von tschechoslowakischen als auch ausländischen Ausstellern wurden CAD- und CAM-Systeme bzw. Komponenten für solche Systeme vorgestellt.

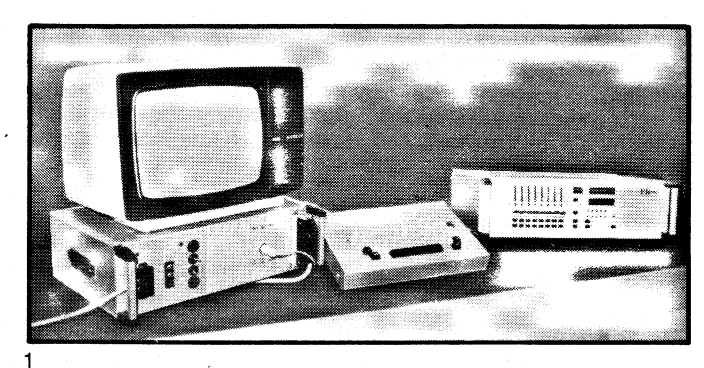
Computertechnik

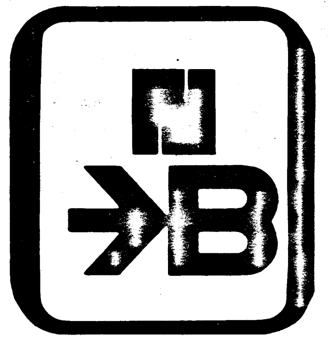
Das sicher interessanteste Exponat der ČSSR-Exposition zur Computertechnik war der **32-Bit-Rechner SM 52/12**, umfangreich konfiguriert mit grafischer Technik. Das Rechnersystem SM 52/12 – mit einer Goldmedaille auf der Messe ausgezeichnet – kann in einem sogenannten „Kompatibilitätsregime" die meisten Instruktionen der 16-Bit-Rechnersysteme SM 3–20 und SM 4–20 abarbeiten. Die technischen Parameter des SM 52/12 sind auf der 3. Umschlagseite ausführlicher dargestellt. Eine Weiterentwicklung des 8-Bit-Mikrorechnersystems SM 50/40–1 ist das **16-Bit-Mikrorechnersystem M 16–1**. Das Mikrorechnersystem von ZVT Banská Bystrica basiert auf dem Mikroprozessor K 1810 aus der UdSSR. Die Hauptspeicherkapazität beträgt 1 MByte. Die maximal mögliche Speicheradressierung ist 16 MByte. Das **16-Bit-Mikrorechnersystem M 16–22** (auch als SM 50/50.M1 bezeichnet) ist eine Nachfolgeentwicklung des bekannten Systems SM 50/50. Die Vorteile des M 16–22 gegenüber dem SM 50/50 sind u. a. die Möglichkeit der Mikroprogrammierung durch den Nutzer, ein Gleitkommaprozessor sowie eine maximale Hauptspeicherkapazität von 2 MByte (Standardausführung 0,5 MByte). Die Produktion soll bereits 1986 begonnen haben. Mit grafischen Bildschirmen CM 7202.M2 und einer entsprechenden Software läßt sich der Computer als CAD-System einsetzen.

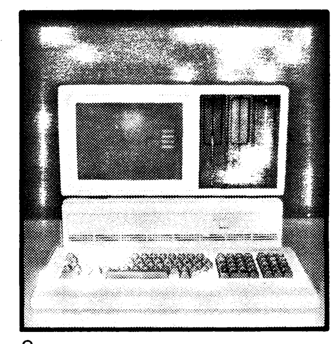
MIKROPAS 80 (Bild 1) ist ein auf dem Prozessor MHB 8080A basierendes universelles Mikroprozessor-Baukastensystem, welches sich vor allem für einfache und mittlere Applikationen für das Steuern, Messen, Regeln, für kleinere DV-Aufgaben und als intelligente Peripherie in größeren Systemen eignet. Die hohe Universalität des Systems ergibt sich aus der Möglichkeit, eine Vielzahl von Standardmoduln entsprechend den Anforderungen zu konfigurieren. Als neueste Module wären z. B. zu nennen: EPROM-Programmiermodul PROGR, 12-Bit-A/D-Wandler, 12-Bit-D/A-Wandler, Modul für serielle Datenübertragung, 8 × 2 KByte-EPROM und Modul für parallele Datenübertragung mit Optokopplern.

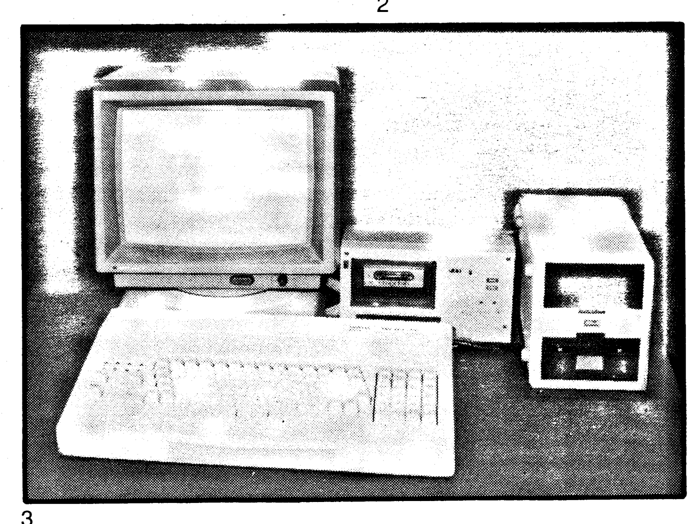
Großes Angebot an Personalcomputern

Neuigkeiten gab es ebenfalls in der Personalcomputer-Leistungsklasse. Beispielsweise das **Datenerfassungssystem CONSUL 2715** (Bild 2) mit 192 KByte Operationsspeicherkapazität und bis zu 3 Diskettenlaufwerken des Typs CONSUL 7115 mit einem Speichervermögen von bis zu 3,6 MByte. Software für die Datenverarbeitung, z. B. Sortier- und Mischfunktionen, werden ebenso wie

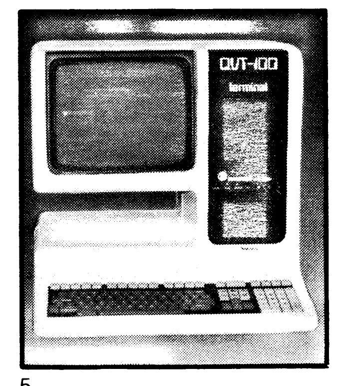

1




2


3


4



5

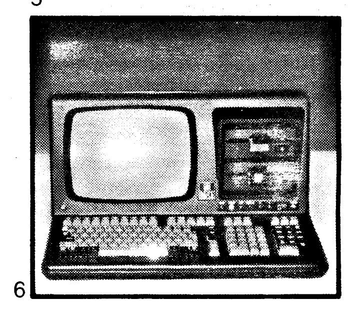

6

Bild 1 Mikroprozessor-Baukastensystem MIKROPAS 80
Bild 2 Datenerfassungssystem CONSUL 2715
Bild 3 Personalcomputer PP 01.16
Bild 4 Einkartenrechner PC-88
Bild 5 Terminal QVT-100 für PC TPA-Quadro
Bild 6 Intelligentes Terminal M3T 320.4

Textverarbeitungsprogramme zur Verfügung gestellt. Der PC **PP 01.16** (Bild 3) von VÚVT Žilina hat einen 8088-äquivalenten Mikroprozessor. Als RAM-Speicher sind 64 KByte, aufrüstbar bis 320 KByte angegeben. Im 40-KByte-ROM ist u. a. der BASIC-Interpreter abgespeichert. Der grafische Monitor hat ein Maximalraster von 640 mal 200 Punkten. PP 01.16 ist mit einem Magnetbandkassettengerät betreibbar. Steht eine Disketteneinheit zur Verfügung, ist das Betriebssystem PP DOS mit ASSEMBLER, PASCAL und FORTRAN nutzbar.

Der **PP-06** aus Banska Bystrica hat ebenfalls den Prozessor I 8080 und den I 8087 als mathematischen Koprozessor sowie 256 KByte RAM und 40 KByte ROM. Es ist grafische und semigrafische Arbeit möglich. An Betriebssystemen sind PP DOS mit den Sprachen PASCAL, COBOL, FORTRAN, BASIC und PL-1 sowie MIKROS 86 (kompatibel zu CP/M 86) implementiert.

Den **Einkartenrechner PC-88** (Bild 4) hatte Tesla Piestany vorgestellt. Auch hier bildet ein 8088-äquivalenter Prozessor die Grundlage. 192 KByte RAM und 8 KByte ROM hat der PC-88 in der Grundversion. Der Monitor ist über RGB-Eingang angeschlossen. Im alphanumerischen Modus können einfarbig 80 × 25 Zeichen und mehrfarbig 40 × 25 Zeichen dargestellt werden – im Grafik-Modus 640 × 200 Punkte einfarbig und 320 × 200 Punkte mehrfarbig. Der PC-88 ist mit einem Centronics-Interface ausgerüstet.

Gleich 4 Prozessoren soll der PC **TPA-Quadro** vom Zentralforschungsinstitut für Physik der Ungarischen Akademie der Wissenschaften haben. Die Größe des Arbeitsspeichers ist mit 128 K Worten angegeben. Wahlweise ergänzen zwei bis vier Diskettenlaufwerke das Gerät. Maximal 12 Terminals sollen an den PDP-8-kompatiblen PC angeschlossen werden können. Bild 4 zeigt ein Terminal QVT-100 für den TPA-Quadro.

Das Unternehmen Metra Blansko aus der ČSSR zeigte aus seiner Terminalreihe M3T 320.x ein neues Gerät – das **intelligente Terminal mit Grafikdisplay M3T 320.4** (Bild 6). Wichtigste Merkmale sind: 16-Bit-Mikroprozessor, 64 KByte EPROM, 16 KByte RAM, zwei 5¼-Zoll-Floppy-Disk-Laufwerke, residenter BASIC-Interpreter, ASSEMBLER und Grafikprogramme. Auf dem Bildschirm lassen sich in 32 Zeilen je 80 Zeichen darstellen. Die Auflösung beim Grafikmodus beträgt 640 mal 368 Rasterpunkte.

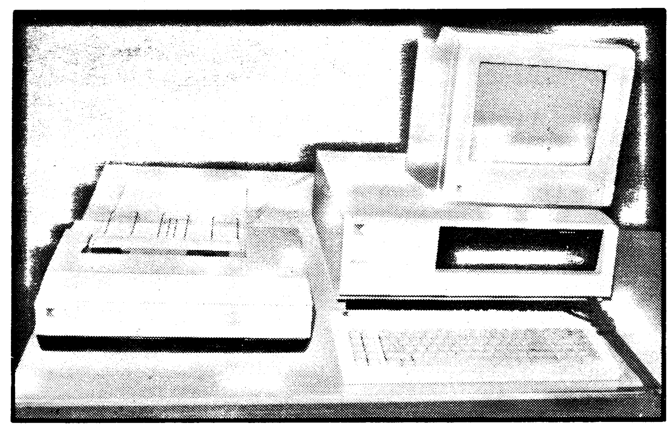
Das polnische Außenhandelsunternehmen Metronex war mit einem breiten Hardware-Spektrum vertreten, vom Matrixdrucker D 100 E bis zur Dispatcherzentrale. Hervorzuheben ist der IBM-PC-kompatible Personalcomputer **Mazovia 1016** (Bild 7) mit dem 16-Bit-Prozessor K 1810 WM 86 und einem mathematischen 8087-Koprozessor. Der RAM beträgt 256 bis 640 KByte, der ROM 48 KByte. Als externe Speicher stehen Floppy-Disk von 2 × 360 KByte oder 2 × 180 KByte sowie Festplatten mit 10 bis 20 MByte zur Verfügung. An Betriebssystemen werden PC DOS, CP/M 86 und XENIX angeboten, Compiler existieren für BASIC, PASCAL, FORTRAN, COBOL, C und Makroassembler.

1987 in die Serienproduktion gehen soll ein neuer, bei Tesla vorgestellter Heimcomputer: **Ondra** (Bild 8), basierend auf dem Mikroprozessor U 880 D. Das Gerät erlaubt die Schwarzweißdarstellung von Groß- und Kleinbuchstaben und ist grafikfähig. Die Hauptspeicherkapazität beträgt 16 bzw. 64 KByte, davon 10 KByte Bildspeicher; der EPROM-Festwertspeicher hat 4–16 KByte. Als Software stehen zunächst ein 4-KByte-Monitor und ein 11-KByte-BASIC-Interpreter (auf Kassette) zur Verfügung, vorgesehen sind MIKOS (kleines Kassetten-Operating-System), TOOL (Assembler, Disassembler), TEDIT (Text-Editor) und die volle Version von BASIC G.

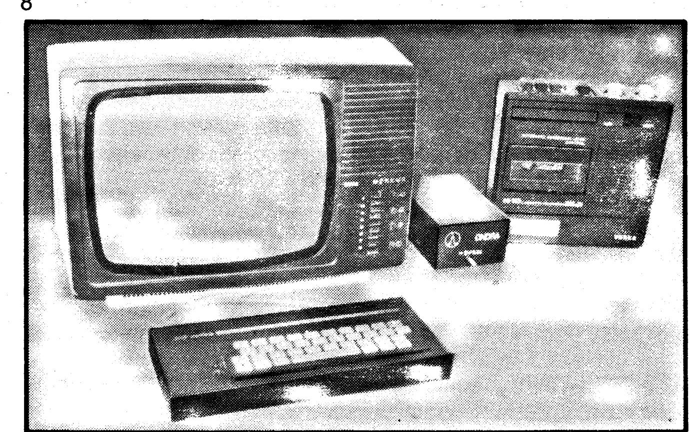
Eine Reihe namhafter internationaler Konzerne auf dem Gebiet der Computertechnik offerierte ihre Erzeugnisse in erster Linie in Halle D. Hier können beispielsweise Firmennamen wie IBM, NCR, ICL, Wang und Olivetti genannt werden. Mit einigen, z. B. Olivetti und NCR, arbeitet der tschechoslowakische Betrieb datasystem auf dem Gebiet der Softwareerstellung zusammen. Olivetti stellte u. a. den **P. E. (Personal Engineering) 24** (Bild 9) als grafische CAD/CAM-Station aus. Prozessor 16-Bit-8086 (10 MHz), arithmetische Einheit 8087, 640 KByte RAM, 16 KByte ROM sind die wichtigsten Charakteristiken. 360-KByte-Floppy- und 20-MByte-Hard-Disk-Einheit sind in das Gerät integriert. Als Betriebssysteme werden MS DOS und XENIX, als Programmiersprachen FORTRAN, PASCAL, C sowie BASIC genannt.

Der **PC 6** (Bild 10) von NCR – kompatibel mit IBM-PC/XT – besitzt den Mikroprozessor I 8088-2 (8 MHz). Der RAM-Speicher ist je nach Modell bis auf 640 KByte erweiterungsfähig. Ebenfalls abhängig vom Modell ist die Ausstattung mit ein oder zwei Diskettenlauf-

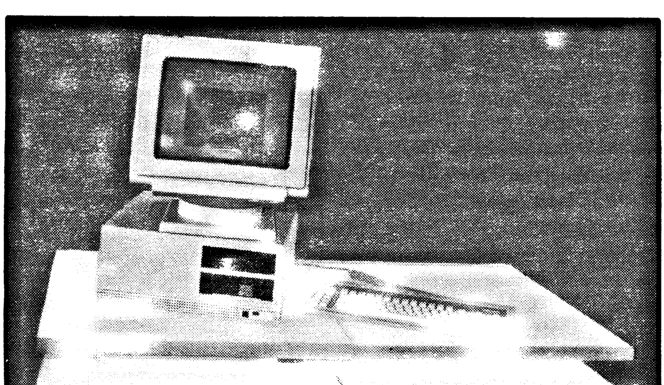
7



8



9



10

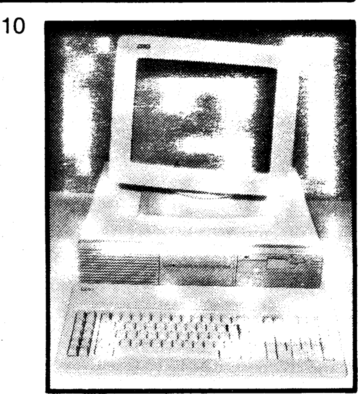


Bild 7 Personalcomputer Mazovia 1016
Bild 8 Heimcomputer Ondra
Bild 9 CAD-Station P. E. 24
Bild 10 PC 6

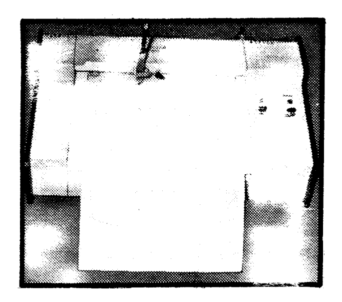
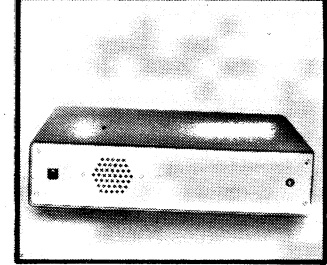
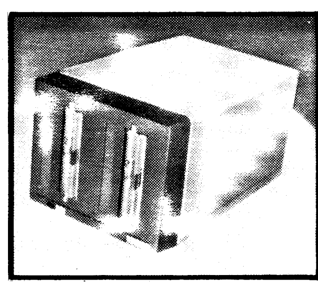

11

Bild 11 Plotter XY 4131
Bild 12 Plotter MERA 621 am 16-Bit-PC MERA 660
Bild 13 Sprachausgabegerät HV 01
Bild 14 Diskettenstation PFD 251
Bild 15 CNC-Bohrmaschine ARITMA 1701
Bild 16 Laserschneidsystem LCS 400-1
Fotos: Paszkowsky (9), Weiß (7)


13


14


12


15

16

werken zu je 360 KByte bzw. ein oder zwei Winchestereinheiten zu je (maximal) 20 MByte.

Peripheriegeräte

Ein neuer tschechischer Plotter hauptsächlich für Mikrorechner der unteren Leistungsklasse ist der **XY 4131** (Bild 11) für Zeichnungen im A4-Format. Es sind die Zeichnungsgeschwindigkeiten 40, 60, 80 und 100 mm pro Sekunde programmierbar. Der Plotter wiegt etwa 5 kg.
Eine interessante Neuentwicklung stellt auch der in Gemeinschaftsarbeit zwischen der VRP und der ČSSR entstandene Plotter **MERA 621** dar (Bild 12). Er verfügt über das Format A3, 4 Farben, einen Mikroprozessor UCY 7880 (8080A) und die Hewlett-Packard-Grafiksprache HPGL. Weitere Daten: max. Genauigkeit 0,05 mm, max. Geschwindigkeit 300 mm/s, Beschleunigung bis 2 g, Schnittstelle V.24.
Die Signale einer Schwarz/weiß- oder Farbfernsehkamera werden vom **interaktiven Videokoprozessor DIGITES 2 C** in eine für die weitere Verarbeitung durch Mikrorechner geeignete Form umgewandelt. Dabei ist durch den Bildschnellspeicher des Gerätes die Aufzeichnung von 128 mal 128 Bildpunkten mit 16 Grauwertstufen (16 K mal 4 Bit) möglich, z. B. für die Verarbeitung von Röntgenaufnahmen. Für viele Einsatzzwecke ist die Unterscheidung in Grauwertstufen nicht notwendig, z. B. bei der Lagebestimmung des Greifarmes eines Roboters. Dann können die Daten im Maßstab 256 mal 256 mal 1 Bit geschrieben werden. Das Gerät ist an Computersysteme der SKR-Reihe anschließbar. Texte in tschechischer Sprache gibt das **Sprachausgabegerät HV 01** (Bild 13) von Tesla Rožnov aus. HV 01 basiert auf dem Mikroprozessor U 880 D. Der Synthesator wird als Peripheriegerät an ein Rechnersystem angeschlossen. Im Rechnersystem müssen die auszugebenen Daten in geeigneter Form abgespeichert sein. Die Silben-, Wort- und Zahlenverständlichkeit betragen 75, 95 und 100 Prozent.
Der Betrieb Pragotron zeigte eine für den Schul-Mikrorechner IQ 151 entwickelte – aber auch an anderen Kleincomputern nutzbare – Diskettenlaufwerkseinheit (Bild 14), bestehend aus einem oder zwei Laufwerken (Typ **PFD 151** bzw. **PFD 251**). Verwendet werden einseitige 8-Zoll-Disketten mit einfacher Aufzeichnungsdichte. Das Format beträgt 77 Spuren × 26 Sektoren × 128 Byte; die Diskettenkapazität 3,1 bzw. 6,2 MBit. Die 330 × 288 × 560 mm^3 große Einheit wiegt 22 bzw. 30 kg.
Die **CNC-Bohrmaschine ARITMA 1701** (Bild 15) für das Ausbohren von Löchern in Leiterplatten erhielt Messegold. Die Maschine wird durch ein Multiprozessorsystem auf Basis des Mikrorechnersystems SM 50/40 gesteuert. Die Anlage verfügt über drei Bohrspindeln und ermöglicht so das gleichzeitige Ausbohren von 3 Leiterplatten. Nicht die Bohrspindeln werden bewegt, sondern die Arbeitsplatte auf einem Luftpolster.
Das Steuerprogramm für ARITMA 1701 erhält man entweder als Teilergebnis des automatisierten Plattenentwurfs oder unter Nutzung der speziellen Sprache COTEL. Die Befehle sind in Bewegungs-, technologische, Hilfs- und Unterprogramminstruktionen unterteilt. Die Koordinaten X und Y können absolut sowie inkremental programmiert und in μm oder in einem selbstgewählten Raster angegeben werden. Die Dateneingabe erfolgt über Lochband oder Diskette.
Eine Gemeinschaftsentwicklung zwischen mehreren Betrieben und Institutionen der DDR und der ČSSR ist das CNC-gesteuerte **CO_2-Laserschneidsystem LCS 400-1** (Bild 16). Verwendet der 400-Watt-Laser SM 400 mit einer optischen Ankopplung. Damit lassen sich die verschiedensten nichtmetallischen Werkstoffe wie Holz, Plast, Asbest, zum Teil auch Keramik schneiden und gravieren.
Bemerkenswert ist, daß für die off-line-Steuerung des Lasers das Steuersystem des Plotters Digigraf 1208/3.5G in modifizierter Form verwendet wird.
Für den on-line-Betrieb ist der Einsatz verschiedener Minirechner, zum Beispiel SM 4–20, SM 50/50, SM 52/11 oder ADT 4500, möglich.

I. Paszkowsky, H. Weiß

Wissensspeicher Mikrorechnerprogrammierung

Reihe Technische Informatik, von L. Claßen und U. Oefler, VEB Verlag Technik Berlin 1986, 200 S.

Bereits in der zweiten Auflage erschien 1986 das Fachbuch mit dem Titel: Wissensspeicher Mikrorechnerprogrammierung. Diese Tatsache bestätigt das dringende Bedürfnis, das in der Industrie wie im Hoch- und Fachschulwesen an diesem sehr effektiven Informationsmaterial besteht. Der Titel des Buches drückt den Inhalt sehr ungenau aus, denn Gegenstand der Beschreibungen sind die Mikroprozessorschaltkreisfamilien (etwa 60 Prozent), nicht aber Mikrorechner. So ist das Assemblerniveau einer Programmiersprache immer mehr nur eine Maschinen-(Prozessor-)Sprache, nicht aber die Sprache eines Rechners.
Damit wendet sich das Buch inhaltlich zu allererst an jene, die Mikrorechner entwerfen und weniger an jene, die sie nutzen – auch wenn mit den Abschnitten U 880-BASIC- und PASCAL-Interpreter sowie U 883-TINY-MPBASIC auf die Mikrorechnernutzung orientiert wird. Aber auch hier bleibt die Bindung an die Prozessoren erhalten. Die beschriebenen Programmiersprachen laufen nur unter dem Betriebssystem UDOS (RIO), was auf den Entwicklungssystemcharakter des zugrundegelegten Mikrorechners hinweist. Denn nur für diese Ausgabe ist das Betriebssystem UDOS relevant.
Damit erreicht das Buch eine nützliche spezifische innere Geschlossenheit. Die drei behandelten Mikroprozessortypen U 880 (Z80), U 881/882 (Z8) und U 8000 (Z8000) sind in der DDR dominierend, auch wenn sie international kaum den Einzug in Massen-Mikrorechnern (Personalcomputer, PC) gehalten haben. Überwiegend findet man sie in Steuerrechnern für industrielle Anwendungen. Ganz eindeutig ist dies bemerkenswerterweise beim Z 8000, wogegen es für den Z 8 von vornherein selbstverständlich ist. Bezüglich der Prozessorbeschreibung erfaßt das Buch nur die Programmiereigenschaften, nicht aber die Signaleigenschaften. Das ist entsprechend der Zielsetzung des Buches zur Programmierung beabsichtigt.
Der Nutzer des Buches stellt schnell fest, daß die Autoren eine gute Sachkenntnis im Detail besitzen. Der Aufbau des Stoffes ist folgerichtig, logisch und verständlich mit effektiven Übersichten, wobei die Beschränkung auf das Wesentliche eher nützlich als nachteilig ist.
Die Reihe Technische Informatik – in der Herausgabe von Dr. Ludwig Claßen, Dr. Dieter Nedo und Dr. Gerhard Paulin – hat mit dem ersten Band eine richtige Titelwahl getroffen.
Man kann den Verlag, die Herausgeber und die Autoren dazu beglückwünschen.
Ausstattung des Buches und Preis sind angemessen. Ebenso ist es zu wünschen, daß in dieser Reihe viele Titel erscheinen.

Prof. Dr. M. Roth

Das Hochgeschwindigkeits-Schaltkreissortiment K 1800

von W.-B. B. Arbaitis, S. Ju. Sedauskas, A.-W. W. Pjatrauskas
Verlag „Radio i Swjas", Moskau 1986, 184 S.

In der Broschüre werden die konstruktiv-technologischen, schaltungstechnischen und strukturellen Besonderheiten der ECL-LSI-Schaltkreisfamilie K 1800 behandelt. Dabei handelt es sich innerhalb des Typenspektrums von Prozessorschaltkreisen mit der Bezeichnung K 18xx um Hochgeschwindigkeits-IS mit einer Verarbeitungsbreite von 4 Bit, die mit Taktfrequenzen von 36 MHz (Zykluszeit 40 ns) betrieben werden. Die Familie K 1800 enthält u. a. die Funktionseinheiten ALU (28 logische, 23 binäre, 17 BCD- sowie logische und arithmetische Verschiebeoperationen), Mikroprogrammsteuereinheit, Speichersteuerung, Synchronisations-IS, ECL-TTL-Koppler, Pufferspeicher sowie Zusatzschaltkreise.
In der Broschüre wird zunächst eine Einordnung der Familie K 1800 in die Serien K 5xx und K 18xx vorgenommen. Außerdem wird ein Überblick über die wichtigsten Eigenschaften und strukturellen Besonderheiten gegeben. Anschließend werden die einzelnen Schaltkreise ausführlich vorgestellt (Anschlußbild, Blockschaltbild, technische Daten, Befehlsübersicht, Funktionsbeschreibung, Anwendungsaspekte). Es folgt eine genau und detaillierte Zusammenstellung aller technischen Daten der IS, wobei auch auf meßtechnische Probleme eingegangen wird. Im 5. Kapitel werden Anwendungsaspekte behandelt (Erhöhung der Verarbeitungsbreite, Programmbeispiele, Anwendungsbereiche, Geschwindigkeitsverhalten von Rechnern auf Basis K 1800).
Die Schaltkreisfamilie K 1800 wird anstelle der älteren Familie K 500 eingesetzt und bringt (bei 1/3 bis 1/5 des Realisierungsaufwandes gegenüber K 500 – Lösungen) Verarbeitungsgeschwindigkeiten um 1 Million Befehle je Sekunde. Gut geeignet sind die Schaltkreise u. a. für die Realisierung von Spezialprozessoren (z. B. für den Einsatz in lasergestützten Bilderkennungssystemen, Massenspektrometern oder bei der schnellen Fouriertransformation). Für Entwurf, experimentelle Erprobung und Realisierung von Spezialprozessorarchitekturen (z. B. KI- oder HLL-Prozessoren) ist die Mikroprogrammierbarkeit eine wichtige Eigenschaft. Dem Anwender von Schaltkreisen der Serie K 1800 werden mit dieser Broschüre umfassende und vollständige Informationen für seine praktische Arbeit gegeben.

G.-U. Vack

INFO 88

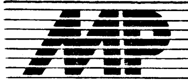
Die Gesellschaft für Informatik der DDR führt gemeinsam mit Einrichtungen der Akademie der Wissenschaften und des Hochschulwesens, mit den Kombinaten Robotron, Datenverarbeitung und Nachrichtenelektronik und der WGMA der KDT vom 22. bis 27. Februar 1988 an der Technischen Universität Dresden die INFO 88 als nationalen Kongreß der DDR-Informatiker durch. Anliegen dieser Veranstaltung ist es, den aktuellen Leistungsstand der Informatik als Schlüsseltechnologie bei der breiten Rationalisierung unserer Volkswirtschaft – von den Grundlagen bis zu den Anwendungen – zu demonstrieren, über den Stand der Forschung und Trends auf Schwerpunktgebieten zu informieren und den Entwicklungsstand von Teilgebieten der Informatik darzustellen.

Die Tagung erfolgt im Plenum und in Fachsektionen zu folgenden Themenkomplexen:

1. **Theoretische Grundlagen** mit algebraischen und logischen Grundlagen der Programmierung, Kompliziertheitstheorie und innovativen Algorithmenkonzepten, mathematischen Grundlagen der künstlichen Intelligenz und des Datenbankentwurfs
2. **Computertechnik** mit Computerarchitekturen, Computersystemen, Spezialprozessoren sowie Computernetzen (lokale, Weitverkehrsnetze, Einsatz und Betrieb)
3. **Software** mit rechnergestützter Entwicklung und Produktion, sprachlichen Beschreibungs- und Programmierwerkzeugen, Programmier- und Datenbanktechniken
4. **Künstliche Intelligenz** mit natürlichem Sprachverstehen, KI-Sprachen, Wissensverarbeitung und Expertensystemen, kognitiven Systemen
5. **Komplexe Anwendungen** mit CAD, CAM, CIM; Bildverarbeitung, Informationsbeziehungen im Büro, Modellierung und Verhaltenssimulation komplexer Systeme, Experimentenautomatisierung, automatisierten Informationsdiensten in Wissenschaft und Technik
6. **Aus- und Weiterbildung** mit Informatik-Grundausbildung in Schulen, Berufs- und Hochschulen, Weiterbildung der Werktätigen, Computer im Bildungswesen
7. **Gesellschaft und Informatik** mit Informationstechnik und Information als Faktor der intensiv erweiterten Reproduktion, Gestaltung informationeller Prozesse in der sozialistischen Volkswirtschaft, Einfluß der Informatik auf das System der Wissenschaften.

Vortragsangebote (Kurzfassung bis zu 3 Seiten) sind bis zum 1. März 1987 an das Sekretariat der GIDDR, Clara-Zetkin-Str. 105, Berlin, 1086 zu senden.

Prof. Dr. D. Hammer, Vorsitzender der GIDDR und des Programmkomitees / Prof. Dr. H. Tzschoppe, Vorsitzender des Vorbereitungskomitees



Dokumentation

Sorst, M.; Gieseler, M.; Fischer, W.-J.:
CMOS-Gate-Array-System U5200 – Schaltungstechnik und Anwendung
Mikroprozessortechnik, Berlin 1 (1987) 1, S. 4
Es werden der Zusammenhang zwischen der Konstruktion der Gate-Array-Zelle und den elektrischen Parametern der Macros, schaltungstechnische Lösungen für ein prüffreundliches Interface und ein LSSD-Master-Slave-Flip-Flop, der Einfluß der schaltungs- und layouttechnischen Gestaltung des zentralen Taktsystems auf den Taktskew sowie die unterschiedlichen Möglichkeiten für die Nutzung des Taktes beim Aufbau synchroner Schaltungen beschrieben.

Sorst, M.; Giseler, M.; Fischer, W.-J.:
МОП-система U 5200 на основе вентильных матриц – схемотехника и применение
Mikroprozessortechnik, Berlin 1 (1987) 1, стр. 4
Описаны связь между конструкцией элемента на основе вентильных матриц и электрическими параметрами макроса, схемитехнические решения для приятного для испытания интерфейса и двухступенчатого триггера LSSD, влияние схемотехнического и топологического оформления центальной тактирующей схемы на перекос такта, а также различные возможности для пользования тактом при построении синхтонных схем.

Sorst, M.; Giseler, M.; Fischer, W.-J.:
CMOS Gate-Array System U5200 – Circuitry and Application
Mikroprozessortechnik, Berlin 1 (1987) 1, pp. 4
The authors describe the relation between the construction of the gate array cell and the electrical parameters of the macros, the circuit solutions for an interface easy to test and a LSSD master-slave flip-flop, the influence of the circuit and layout of the central clock system upon the clock skew as well as the different possibilities for using the clock with the design of synchronous circuits.

Bala, P.; Haupt, R.; Claßen, L.:
Das Echtzeitbetriebssystem IRTS 8000
Mikroprozessortechnik, Berlin, 1 (1987) 1; S. 8
Das Echtzeitbetriebssystem IRTS 8000 ist ein durch den Anwender konfigurierbares Standard-Softwaresystem für Echtzeit-Mikrorechner. Es unterstützt die 16-bit-Mikroprozessorfamilie U8001/U8002 und besteht aus einem relativ kleinen Systemkern von 4 kByte, um den anwendungsspezifische Ergänzungsmodule – wie eine formatgesteuerte Ausgabeorganisation, Terminal-/Druckerhandler, Testhilfsmittel u. a. m. – angeordnet werden können.

Bala, R.; Haupt, R.; Claßen, L.
Операнионная система реального времени IRTS 800
Mikroprozessortechnik, Berlin, 1 (1987) 1, стр. 8
Операционная система реального времени IRTS 8000 является конфигурируемой потребителем стандартной системой математического обеспечения дла микровачислителей реального времени. Она поддерживает семейство 16-тиразтядных микропроцессоров U 800/U 8002 и состоит из относительно небольшого ядра системы 4 кбайт, вокруг которого могут быть расположены специфические по прикладному программированию дополняющие модули – форматоуправляемая организация выдачи, терминальные и печатающие средства манипулирования, испытательные средства и др. –.

Bala, P.; Haupt, R.; Claßen, L.:
The Real-Time Operating System IRTS 8000
Mikroprozessortechnik, Berlin, 1 (1987) 1, pp. 8
The real-time operating system IRTS 8000 represents a standard software system for microcomputers working in real-time and can be configurated by the user. It supports the 16-bits microprocessor family U8001/U8002 and consists of a relatively small system nucleus of 4 kBytes which is extensible by application-specific modules such as format-controlled output organization, terminal and printer handler, and debugging aids.

Seifart, M.
Intelligenter Prozeßkoppelmodul
Mikroprozessortechnik, Berlin, 1 (1987) 1, S. 11
Der Beitrag beschreibt einen universell einsetzbaren programmierbaren Prozeßkoppelmodul (PPM), der mit 7 analogen/binären Eingängen, einem Zähleingang und 7 Ausgängen ausgestattet ist. Charakteristisch für diesen PPM sind der zusätzliche, galvanisch getrennte Feldbusanschluß, seine umfangreiche Programmierbarkeit und Datenvorverarbeitung sowie die äußerst flexible Anpassung an eine Vielzahl unterschiedlicher Instrumentierungsaufgaben.

Seifart, M.
Интеллектуальный модуль связи с процессом
Mikroprozessortechnik, Berlin 1 (1987) 1, стр. 11
Статья описывает универсально применяемый программируемый модуль связи с процессом, оборудованный 7 аналогово-двоичными входами, 1 входом счета и 7 выходами. Характерными для данного модуля связи с процессом являются дополнительное соединение шин поля с гальванической развязкой, его широкая программируемость и предварительная обработка данных, а также очень гибкое согласование со множеством различных задач инструментации.

Seifart, M.:
Intelligent Process Coupling Module
Mikroprozessortechnik, Berlin, 1 (1987) 1, pp. 11
The author describes a programmable process coupling module which is universally applicable and provided with seven analog/binary inputs, one counting input, and seven outputs. The additional non-galvanic field bus connection, the extensive programmability and data preprocessing a well as the highly flexible fitting to a multitude of different instrumentation tasks are characteristic of this process coupling module.

Horn, Th.
Programmierung in C
Mikroprozessortechnik, Berlin 1 (1987) 1, S. 15
Die Programmierprache C stellt eine moderne höhere Programmiersprache dar, die auf Grund ihrer Portabilität, Flexibilität und Effektivität insbesondere auf 16- und 32-bit-Rechnern, aber auch auf 8-bit-Rechnern, heute eine weite Verbreitung gefunden hat. Die mit diesem Beitrag (Einführung in die Lexik, einfache Datentypen und Speicherklassen) eingeleitete Artikelreihe soll dem Leser eine Einführung in die Anwendung von C geben.

Horn, Th.
Пропраммирование на языке C
Mikroprozessortechnik, Berlin 1 (1987) 1, стр. 15
Язык программирования C представляет собой современный язык высокого уровня, который на основе своей портативности, гибкости у эффективности, в частности на 16-ти и 32-хразрядных вычислителях, а также на 8-миразрядных вычислителях, в настоящее время имеет широкое распространение. Данная первая статья (введение в лексику, простые типы данных и классы памяти) целого ряда статей должна показать читателю введение в применение языка C.

Horn, Th.
Programming in C
Mikroprozessortechnik, Berlin 1 (1987) 1, pp. 15
The programming language C presents A modern high-level programming language today having a wide-spread use particularly with 16- and 32-bits computers but with 8-bits computers, too, due to its portability, flexibility, and efficiency. The series of publications started by the present contribution (introduction to the vocabulary, simple data types, and memory classes) is to introduce the reader to the application of C.

Herden, D.; Lüdicke, R.; Wippich, C.
Semigrafik für PC 1715
Mikroprozessortechnik, Berlin, 1 (1987) 1, S. 19
Im Beitrag wird eine Möglichkeit der grafischen Informationsverarbeitung auf dem PC 1715 mit Hilfe einer Option (nachrüstbare Leiterplatte) und eines Grundsoftwarepaketes dargestellt. Das Grundsoftwarepaket GEDIT wurde in der Form eines allgemeinen grafischen Editors konzipiert, um dem Nutzer eine interaktive Erstellung beliebiger Zeichen, Symbole und Bilder auf dem Monitor ohne Kenntnisse einer Programmiersprache zu ermöglichen. Über definierte Programm- und Datenschnittstellen ist eine Einbindung des Grafikeditors in Anwenderprogramme gegeben.
Der Grafikeditor wurde in Turbo-Pascal programmiert.

Herden, D.; Lüdicke, R.; Wippich, C.
Семиграфик для персональной ЭВМ 1715
Mikroprocessortechnik, Berlin 1 (1987) 1, стр. 19
В статье изображена возможность графической обработки информации на персональной ЭВМ 1715 с помощью дополнительно вставляемой платы и основного пакета программного обеспечения. Основой пакет программного обеспечения GEDIT был конципирован в виде общего графического редактора, для того чтобы сделать возможной для пользователя интерактивная разработка любых знаков, символов и изображений на мониторе без знаний языка программирования. Через определенные места стыковки программ и данных получено включение графического редактора в программу пользователя. Графический редактор программирован на языке ТУРБО-ПАСКАЛ.

Herden, D.; Lüdicke, R.; Wippich, C.:
Quasigraphics for the PC 1715
Mikroprozessortechnik, Berlin, 1 (1987) 1, pp. 19
The authors describe a possibility for graphical information processing on the PC 1715 by means of an option (additional printed circuit card) and a basis software package. The latter was desogned in form of a general graphic editor in order to enable the user to produce any character, symbol, and image on the monitor in an interactive way and without knowledge of a programming language. The editor programmed in Turbo-PASCAL can be incorporated into application programs via defined program and data interfaces.

Termine

1. Internationale Fachtagung Lichttechnik

WER? Fachverband Elektrotechnik in der KDT, Wissenschaftliche Sektion Lichttechnik
WANN? 18. bis 20. März 1987
WO? Berlin
WAS?
- Neue Entwicklungen u. a. bei Lampen, Leuchten, Zünd- und Vorschaltgeräten
- Rationelle Energieanwendung
- Neuentwicklungen der Licht- und Strahlenmeßtechnik
- Erkenntnisse und Tendenzen der Grundlagenforschung
- Einsatz der Mikroelektronik und Rechentechnik bei der Erzeugnisentwicklung, insbesondere in der Projektierung (CAD)

WIE?
Teilnahmemeldungen schriftlich an:
Kammer der Technik, Präsidium,
Fachverband Elektrotechnik,
Postfach 1315, Berlin, 1086

Hoppe

Vorschau

In Mikroprozessortechnik, 2/1987 können sie u. a. lesen:

- Einführung in die Programmiersprache C, 2. Teil
- Entwicklungsunterstützung für ein 16-Bit-Mehrmikrorechnersystem
- Der Kleincomputer KC 85/3

Bildschirm-Fensterkopien mit dem KC 85/2 (/3)

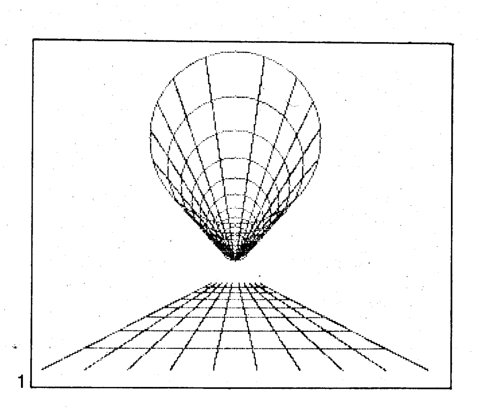
Das Basic der Kleincomputer KC 85/2 und KC 85/3 hat eine Reihe guter Eigenschaften. Eine davon ist die Fenstertechnik, besonders wenn man sie zusammen mit den Grafikmöglichkeiten nutzt. Leider gab es in der vorhandenen Software bisher keine Möglichkeit, solche Fenster abzuspeichern oder auf Papier zu bringen.

Deshalb entstand die im folgenden beschriebene Routine. Das Programm ermöglicht das Abspeichern und Einladen des aktuellen Fensters (Pixel- und Farbspeicher) bis zur Größe der gesamten Bildschirmfläche von Kassette sowie das Ausdrucken auf einem Nadeldrucker (Pixelspeicher), z. B. auf dem K 6313. Mit einem dazu kompatiblen Typ entstand das abgebildete Beispiel (Bild 1).

Das Programm (siehe Hexdump nach Bild 2) besteht aus drei Routinen, die aus dem Menü oder über Basic abrufbar sind: SCRL (Laden), SCRS (Retten) und SCRC (Hardcopy). Das Programm ist im Hexdump mit der Anfangsadresse BA00 versehen, kann aber eine beliebige Anfangsadresse haben. Ebenso kann jede der drei Teilroutinen ausgekoppelt werden. SCRC liegt im Hexdump von BA00 bis BAAE, SCRS von BAAF bis BB5B, SCRS von BB5C bis BBF1.

Aus dem Menü läßt sich eine Routine durch Angabe ihres Namens aufrufen. Aus Basic heraus muß auf den CALL-Befehl zurückgegriffen werden. Die Adresse für den Aufruf einer Teilroutine wird durch Addition von 7 zur Anfangsadresse der Teilroutine ermittelt. Für das Modul-Basic des KC 85/2 und das ROM-Basic des /3, die bei der Ausführung eines Befehls immer den IRM abschalten, kann keine der Routinen direkt aufgerufen werden, da alle auf den IRM zurückgreifen bzw. u. U. selbst im IRM liegen. Deshalb muß für den Ansprung der gewünschten Routine das folgende kleine Programm ab einer Adresse im Bereich von 0 bis 7FFF aufgerufen werden:

CALL F01B CD 1V F0;
Zuschalten IRM
CALL abcd CD cd ab;
Aufruf der Routine ab Aufrufadresse abcd
CALL F01B CD 1B F0;
Abschalten IRM
RET C9; Rückkehr ins Basic

1 

2

```
Hexdump of SCRBASV1. 5/10/86
BA00 7F 7F 53 43 52 43 01 F5  031F
BA08 E5 D5 C5 3A 9D B7 87 87  051B
BA10 87 47 3A 9F B7 87 87 87  03F3
BA18 80 32 93 B7 3E 1B C5 CD  03E7
BA20 BD B7 3E 41 CD BD B7 3E  0472
BA28 08 CD BD B7 3E 00 CD BD  0411
BA30 B7 3E 1B CD BD B7 3E 4B  03DA
BA38 CD BD B7 3A 9E B7 26 00  03F6
BA40 87 87 87 CB 14 E5 CD BD  04E3
BA48 B7 E1 7C CD BD B7 C1 3A  0550
BA50 9C B7 4F 3A 9E B7 81 32  03E4
BA58 92 B7 21 8A B7 3E 08 F5  03E6
BA60 E5 60 69 CD 03 F0 34 56  03F8
BA68 E1 72 23 04 F1 3D 20 EF  03B7
BA70 18 02 18 A8 78 D6 08 57  0287
BA78 1E 08 06 08 21 8A B7 CB  0261
BA80 16 17 23 10 FA C5 D5 E5  03D9
BA88 CD BD B7 E1 D1 E1 1D 20  04F1
BA90 E9 42 0C 79 BE 20 C3 3E  038F
BA98 0D C5 CD BD B7 C1 78 C6  0512
BAA0 08 21 93 B7 BE 28 03 47  02A3
BAA8 18 C8 C1 D1 E1 F1 C9 7F  058C
BAB0 7F 53 43 52 53 01 F5 E5  0395
BAB8 D5 C5 2A 9C B7 7C 87 87  04A1
BAC0 87 67 22 8A B7 22 8E B7  03B8
BAC8 2A 9E B7 7C 87 87 87 67  03F7
BAD0 22 8C B7 22 90 B7 22 00  02F0
BAD8 B7 01 A0 0F DD 36 05 00  027F
BAE0 DD 36 06 B7 CD 03 F0 08  0398
BAE8 21 00 B7 E5 2A 8E B7 CD  03F9
BAF0 03 F0 34 3A 8F B7 1F 38  02FE
BAF8 1C 1F 38 19 1A 22 92 B7  0211
BB00 E1 77 23 CB 7D 28 0A 01  02F6
BB08 40 00 CD 03 F0 01 21 00  0222
BB10 B7 E5 2A 92 B7 7E E1 77  04E5
BB18 23 CB 7D 28 0A 01 40 00  01DE
BB20 CD 03 F0 01 21 00 B7 E5  037E
BB28 2A 8E B7 2C ED 5B 90 B7  042A
BB30 1D 20 20 3A 8A B7 6F 3A  0281
BB38 8C B7 5F 24 15 20 14 01  0210
BB40 40 00 E1 CD 03 F0 01 01  02E3
BB48 40 00 CD 03 F0 09 C1 D1  039B
BB50 E1 F1 C9 22 8E B7 ED 53  0542
BB58 90 B7 18 90 7F 7F 53 43  0383
BB60 52 4C 01 F5 E5 D5 C5 2A  043D
BB68 9C B7 7C 87 87 87 67 22  03ED
BB70 8A B7 22 8E B7 DD 36 05  03C0
BB78 00 DD 36 06 B7 CD 03 F0  0390
BB80 0A 2A 00 B7 22 8C B7 22  0272
BB88 90 B7 CD 03 F0 05 21 00  032D
BB90 B7 E5 2A 8E B7 CD 03 F0  04CB
BB98 34 3A 8F B7 1F 38 19 1F  0243
BBA0 38 16 22 92 B7 E1 7E 12  032A
BBA8 23 CB 7D 28 07 CD 03 F0  035A
BBB0 05 21 00 B7 E5 2A 92 B7  0335
BBB8 D1 1A 77 13 CB 7B 28 07  02EA
BBC0 CD 03 F0 05 11 00 B7 D5  0362
BBC8 2A 8E B7 2C ED 5B 90 B7  042A
BBD0 1D 20 16 3A 8A B7 6F 3A  0277
BBD8 8C B7 5F 24 15 20 0A E1  02E6
BBE0 CD 03 F0 0B C1 D1 E1 F1  052F
BBE8 C9 22 8E B7 ED 53 90 B7  04B7
BBF0 18 A0   00B8
```

Beim Laden wird das Bild an die Anfangsposition des aktuellen Fensters geladen. Es kann über jede beliebige Schnittstelle gedruckt werden. Die Ausgabe eines Bytes an den Drucker erfolgt über USER-Kanal 1 (Basic: USER-kanal 2). Die Ausgaberoutine muß mit Return abgeschlossen sein und darf die Register AF, HL, DE und BC verändern. Die Routine darf keine Veränderung des Byte-Wertes vornehmen (etwa Steuerzeichenausgabe).

Der beste Modus, mit dem die Pixel auf Papier gedruckt werden können, ist der 8-Bit-Nadelmuster-Modus ESC ∗ 5 (576 Pixel) mit einem Zeilenvorschub von 8/72″. Damit wird eine linksbündige Abbildung von Höhe : Breite = 1 : 1 erreicht. Dieser Modus stand allerdings laut Handbuch von 2/85 beim K 6313 noch nicht zur Verfügung. Deshalb wurde mit ESC K gearbeitet, was ein Seitenverhältnis von 1 : 1,2 (Höhe : Breite) bedeutet. Der abgedruckte Hexdump ist auf diesen Modus eingestellt. Der Anwender kann ihn jedoch seinen Möglichkeiten anpassen. Im Hexdump ist in die Speicherzellen BA2D, BA32 und BA37 die Steuerfolge für die Ansteuerung des Nadelmodus einzugeben. Diese Eingabe hat im Falle einer 2-Byte-Steuerfolge (z. B. ESC L) oder bei einer 3-Byte-Folge (z. B. ESC ∗ 5) die Formen:

BA2D 00 (Dummy) bzw. 1B (ESC)
BA32 1B (ESC) bzw. 2C (∗)
BA37 4C (L) bzw. 05 (5 als Wert).

Stefan Schlenzig

Bild 1 Mit dem Programm laut Hexdump vom Bildschirm ausgedruckte Hardcopy eines Grafikfensters auf dem KC 85/2 (/3)

Bild 2 Hexdump für das Ausdrucken von Fenstern vom KC 85/2 (/3) (mit Checksumme)

BA00
BBFF
3x128
384
4x128
BBF1
BA00
1F1
200

ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 1
S. 1–32

Kleincomputer als Prüfbildgenerator

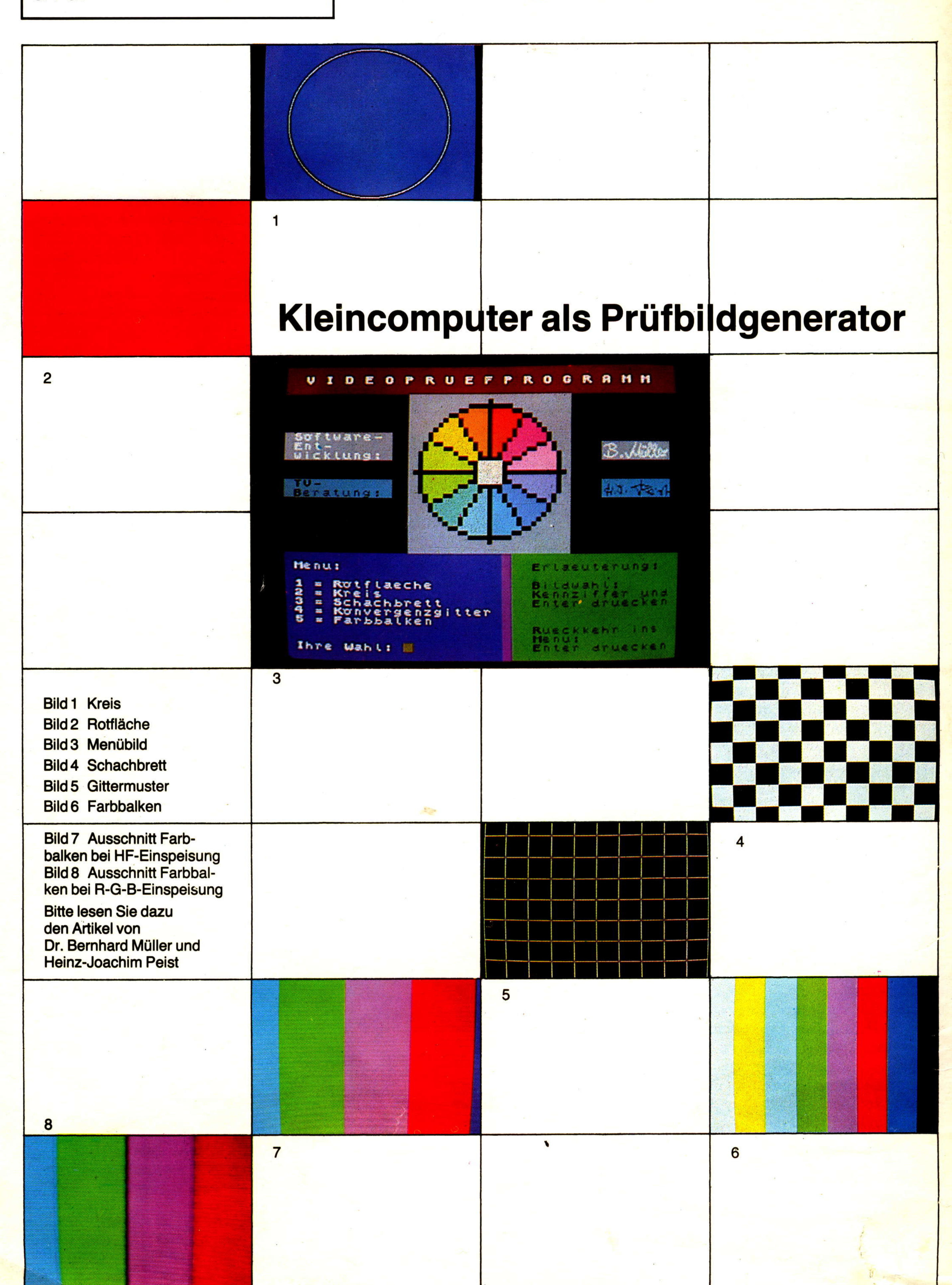


Bild 1 Kreis
Bild 2 Rotfläche
Bild 3 Menübild
Bild 4 Schachbrett
Bild 5 Gittermuster
Bild 6 Farbbalken

Bild 7 Ausschnitt Farbbalken bei HF-Einspeisung
Bild 8 Ausschnitt Farbbalken bei R-G-B-Einspeisung

Bitte lesen Sie dazu den Artikel von Dr. Bernhard Müller und Heinz-Joachim Peist

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 2 · 1987

Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0233-2892

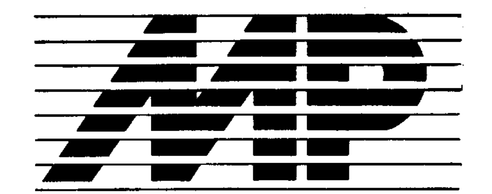
Einchip-Rechner-Schaltkreise

PC in der Meßtechnik

Entwicklungsunterstützung für Mehrmikrorechnersysteme

Kleincomputer KC 85/3 – Hardwarekonzept

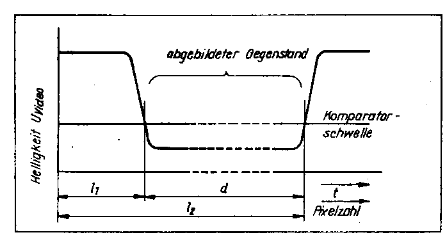
Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0233–2892



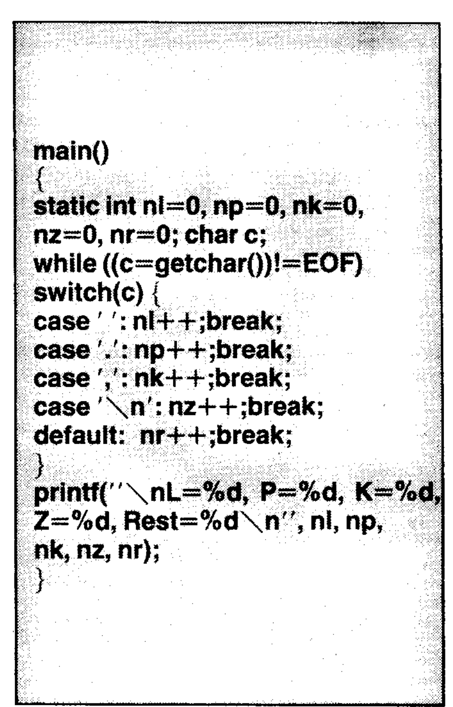
Mikroprozessortechnik, Heft 2 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik
Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR-1020, Berlin; Telegrammadresse: Technikverlag Berlin Telefon: 28700, Telex: 011 2228 techn dd
Verlagsdirektor Klaus Hieronimus
Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat (Tel.: 2870381)
Gestaltung Christina Kaminski (Tel.: 2870288)
Titelbild Tatjana Stephanowitz
Beirat Dr. Ludwig Claßen, Prof. Dr. sc. Dietrich Eckhardt, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Siegmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Jürgen Zaremba
Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik
Gesamtherstellung Druckerei Märkische Volksstimme Potsdam
Erfüllungsort und Gerichtsstand Berlin Mitte.

Redaktionsschluß: 16. Dezember 1986
AN (EDV) 49837
Erscheinungsweise monatlich 1 Heft
Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.
Bezugsmöglichkeiten
DDR: sämtliche Postämter; **SVR Albanien:** Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; **VR Bulgarien:** Direkzia R.E.P., 11a, Rue Paris, Sofia; **VR China:** China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; **ČSSR:** PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; **SFR Jugoslawien:** Jugoslovenska Knjiga, Terazije 27, Beograd; Izdavacko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; **Koreanische DVR:** CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; **Republik Kuba:** Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; **VR Polen:** C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; **SR Rumänien:** D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; **UdSSR:** Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; **Ungarische VR:** P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; **SR Vietnam:** XUNHASABA, 32, Hai Ba Trung, Hanoi; **BRD und Berlin (West):** ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; **Österreich:** Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; **Schweiz:** Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; **Alle anderen Länder:** örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR-7010 Leipzig, und Leipzig Book Service, Talstraße 29, DDR-7010 Leipzig



Seite 41



Seite 47


Seite 60

Foto: Weiß

Inhalt

Vorschau

Für MP 3/87 haben wir für Sie u. a. Beiträge zu folgenden Themen vorgesehen:

- P 8000 Ein universelles 16-Bit-Mikrorechnerentwicklungssystem
- Parallelverarbeitende Rechnersysteme
- Das Softwarekonzept des KC 85/3

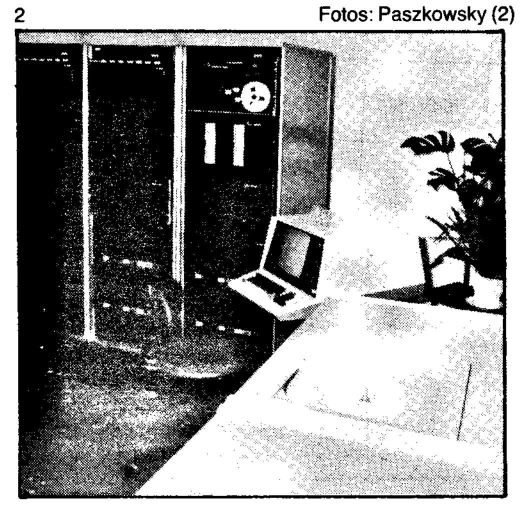
100. SM 4–20 an die DDR übergeben

Am 12. 11. 1986 wurde im Beisein des Stellvertreters des Ministers für Gesundheitswesen, OMR Dr. sc. med. B. Schirmer, des Handelsattachés der ČSSR in der DDR, Gen. Synek, sowie Repräsentanten des Außenhandelsunternehmens der ČSSR Kovo – Export – Import, des Herstellerbetriebes Datasystem Bratislava und des VEB Kombinat Robotron der 100. SKR-Rechner SM 4–20 in der DDR dem Versorgungsdepot für Pharmazie und Medizintechnik Halle in einem festlichen Akt übergeben (Bilder 1 und 2).
Dieser Jubiläumsrechner war zugleich der 25. Rechnér dieses Typs, der seit 1981 dem Gesundheits- und Sozialwesen der DDR übergeben wurde.
Die Rechner dienen zur wirksamen Unterstützung der Leitungs-, Betreuungs- und Versorgungsprozesse. Sie sind in stationären und ambulanten Einrichtungen sowie in allen Versorgungsdepots für Pharmazie und Medizintechnik der DDR eingesetzt.
Der Einsatz der SKR-Rechner im Gesundheits- und Sozialwesen der DDR bedeutet u. a.:
– den schrittweisen Aufbau und die routinemäßige Nutzung von Patienteninformationssystemen in Krankenhäusern und Polikliniken
– die zielgerichtete rechentechnische Bearbeitung einer Anzahl bedeutsamer Forschungsvorhaben des Gesundheitswesens
– den sukzessiven Übergang von der zentralen zur dezentralen Abrechnung der Warenbewegungsprozesse
– die Rationalisierung einer Reihe von Informationsprozessen zur Unterstützung betriebswirtschaftlicher Prozesse
– die Rationalisierung von Leitungs- und Planungsprozessen, einschließlich einer beginnenden Dezentralisierung des umfangreichen Berichtwesens.
Die Nutzung der 25 SKR-Rechner SM 4–20 im Gesundheitswesen kann als ein erster Schritt des Überganges zu einer arbeitsplatznahen und dezentralen Rechentechnik sowie dialogorientierten und prozeßbezogenen Informationstechnologie betrachtet werden, den es durch einen gezielten Einsatz von Personal-, Büro- und Arbeitsplatzcomputern in dieser Fünfjahrplanperiode weiter zu vervollkommnen gilt. Mit dem Einsatz des Rechners wird ein wesentlicher Schritt zur weiteren Intensivierung und Rationalisierung der Leitungs- und Versorgungsprozesse sowie der Transportumschlags- und Lagerwirtschaftsprozesse gewährleistet.
Durch den Hersteller wurde die Übergabe dieses Jubiläumsrechners gerade an den Bereich des Gesundheitswesens als ein Ausdruck des Einsatzes der modernen Rechentechnik für humanitäre und friedliche Zwecke gewürdigt.

P. Straach, D. Witt



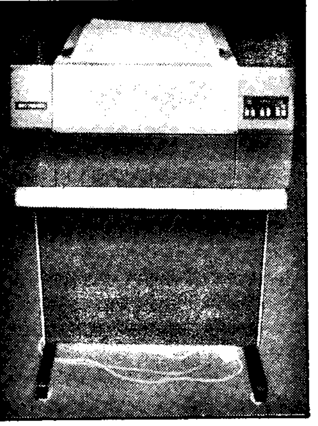
1



2 Fotos: Paszkowsky (2)

Elektrostatischer grafischer Zeilendrucker

Bereits seit längerem bekannt ist der elektrostatische Zeilendrukker für den Anschluß an Computer der SKR-Reihe, EC 7140, aus der ČSSR.
Auf der Internationalen Maschinenbaumesse Brno 1986 wurde nun eine Weiterentwicklung, EC 7240 (Bild 3), vorgestellt. Der EC 7240 ist mit einem Steuerschrank ausgerüstet, der den Anschluß an Rechner des ESER insbesondere an Rechenanlagen EC 1026 und EC 1027 – die in der ČSSR produziert werden



3 Foto: Paszkowsky

ermöglicht. Zum Druck wird elektrografisches Spezialpapier verwendet, dessen dielektrische Schicht eine hohen spezifischen Wiederstand aufweist. Beim Druck bewegt sich das Papier zwischen zwei Elektrodenreihen, die auf der Seite der dielektrischen Schicht aus dünnen Leitern bestehen, deren Anordnung den sogenannten Aufzeichnungskopf bildet. Durch Spannungskombinationen an beiden Elektrodenreihen entsteht in der dielektrischen Schicht ein Bild, das mit Hilfe eines flüssigen Entwicklers im Gerät entwickelt wird. Im wesentlichen sind die Parameter beider Drucker, EC 7140 und EC 7240, identisch. Einige seien hier genannt:
– Druckgeschwindigkeit 1200 Zeilen/min
– Höchstzahl alphanumerischer Symbole in einer Zeile 132
– Punktzahl der grafischen Information in einer Mikrozeile 1320.
Die Ausgabe alphanumerischer und grafischer Informationen ist in einem Modus möglich. Ein auswechselbarer Zeichengenerator ermöglicht den Druck von 4 × 96 grafischen Zeichen.



LAN über das Stromnetz?

Eine kleinere amerikanische Entwicklungsfirma will ein Lokales Netzwerksystem (LAN) entwickelt haben, bei dem auf spezielle Kabel verzichtet werden kann, meldete kürzlich die österreichische Fachzeitschrift Elektronikschau. Zur Übertragung soll vielmehr das vorhandene Stromversorgungsnetz („Steckdose") verwendet werden. Die Idee ist nicht neu. Bisherige Vorstöße in dieser Richtung scheiterten meist am hohen Störsignalpegel im Netz. Bei dem neuen Versuch werden 16 Trägerfrequenzen ab einem Synthesizer benutzt, der zwischen 100 und 460 kHz arbeitet. In der gegenwärtigen Form funktioniert das LAN mit vollem Duplexbetrieb und mit einer Geschwindigkeit von 23 Kbit/s.

Optischer Computer in Sicht?

Der Entwicklungsschritt zum optischen Computer ist wahrscheinlich der amerikanischen Firma telephone and telegraph gelungen. Damit könnten bedeutend mehr Transaktionen gleichzeitig bearbeitet werden als mit der heutigen Datenverarbeitung. Die Firma kündigte an, daß in ihrem Labor eine Art Chip hergestellt worden sei, der die Fähigkeit besitze, Lichtbündel so zu steuern wie in der jetzigen transistorgesteuerten Elektronik elektrische Impulse geregelt werden. Der Chip bestehe nach Angaben des Unternehmens aus Gallium-Arsen- und Aluminium-Gallium-Arsen-Verbindungen, die abwechselnd in mehreren tausend Schichten von jeweils 40 Molekülstärken übereinander geschichtet sind. Unter elektrischer Spannung werden diese Materialien transparent, so daß ein Lichtstrahl sie durchdringen kann. Ein zweiter schwächerer, auf den Chip gerichteter Strahl bewirkt die Neuverteilung der elektrischen Spannung und deren Konzentration in bestimmten Schichten, die dann lichtdurchlässig werden. Der schwächere Lichtstrahl steuert damit die Übermittlung des stärkeren Signalstrahls. Der austretende Lichtstrahl dient gleichzeitig als Input für den folgenden Baustein. Dieser Chip könnte nach Meinung des Unternehmens in einer kommenden Generation von Telefonanlagen einsetzbar sein.

ADN

Softwaretest vor Fertigstellung der dazugehörigen Hardware

Software kann in der Regel erst getestet werden, wenn die dazugehörige Hardware bereitsteht. Um Entwicklungszeit zu sparen, aber auch um Fehlentwicklungen rechtzeitig zu erkennen, wird versucht, die Software schon vor dem Fertigstellen der Hardware auf einer Software-Nachbildung der Hardware, einem sogenannten Modell, zu testen.
Dieses Modell wird durch ein eigenes Computerprogramm erstellt. Die Fa. Siemens entwickelte dafür das Programm Sigmus (Simulator-Generator für Multiprozessorsysteme) speziell für die Modellierung von Multiprozessorsystemen.
Ausgehend von einer Beschreibung der Struktur des Multiprozessorsystems und der einzelnen Komponenten, aus denen es aufgebaut ist, erzeugt Sigmus ein neues Programm – das Modell –, welches das untersuchte Mikroprozessorsystem simuliert. Neben dem Simulator, der den Kern des Modells bildet, wird ein Dialogteil zur Steuerung der Simulation und zur Auswertung der Ergebnisse zur Verfügung gestellt.
Auf dem erstellten Modell des Multiprozessors kann die gleiche Software wie im realen Einsatzfall ablaufen. Auch die entsprechenden Ein- und Ausgabeoperationen sind möglich.

Elektronische Zapfsäule

Zu den Vorteilen der neuentwickelten elektronischen Zapfsäule B 50 P gehört u. a. eine höhere Meßgenauigkeit. Die mikrorechnergesteuerte Zapfsäule, die vom VEB Chemieanlagenbaukombinat Leipzig/Grimma entwickelt wurde, kann alle im Tankwart- und Selbstbedienungssystem anfallenden Daten erfassen und verarbeiten. Die Anlage verfügt über eine doppelseitige LCD-Anzeige sowie über die Anschlußmöglichkeit von Bankschalterterminal mit Tastatur und Bildschirm, Betriebsdatenterminal mit Kartenleser sowie Drukker. Unser Foto zeigt die Neuentwicklung auf der Leipziger Herbstmesse 1986.


Foto: Paszkowsky

12. Mikroelektronik-Bauelementesymposium 1987

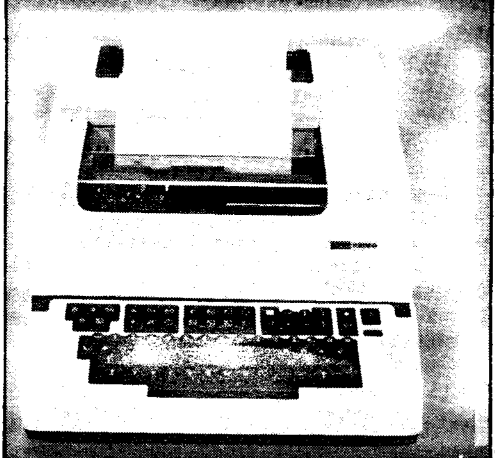
Das 12. Mikroelektronik – Bauelementesymposium des VEB Kombinat Mikroelektronik und des Bezirksvorstandes Frankfurt/Oder der KDT wird unter Schirmherrschaft des Ministers für Elektrotechnik und Elektronik vom 11. bis 16. Mai 1987 wegen des überaus großen Interesses in zwei Teilen durchgeführt.
Dem Symposium vom 11. bis 13. Mai schließt sich unter Leitung des Bezirksvorstandes Frankfurt/Oder der KDT eine Fachtagung vom 14. bis 16. Mai an. Unter dem Motto „Beschleunigte Entwicklung und umfassende Anwendung der Mikroelektronik – Schlüsseltechnologie zur Umsetzung der ökonomischen Strategie des XI. Parteitages" werden die in Entwicklung und Produktion befindlichen mikroelektronischen Bauelemente aus Eigenproduktion und RGW-Import sowie Anwendungsmöglichkeiten vorgestellt. Dieser Zielstellung entsprechen die jeweils (zum Symposium und zur Fachtagung) gehaltenen Fachvorträge, davon Vorträge von Spezialisten aus der UdSSR und ČSSR.
Die inhaltlichen Schwerpunkte orientieren auf
- CAD/CAM-Technik
- moderne Rechentechnik
- elektronische Nachrichtentechnik
- Automatisierung und Robotertechnik, Rationalisierung
- wissenschaftlicher Gerätebau
- Energieerzeugung und -anwendung sowie
- Konsumgüterelektronik.

Im erweiterten Klubprogramm sowie in den Fachdiskussionen werden fachspezifische Probleme behandelt.
Zur Förderung des aktiven Erfahrungsaustausches richten ausgewählte Betriebe aus den Kombinaten Mikroelektronik, Carl Zeiss JENA und Keramische Werke Hermsdorf Konsultationsstützpunkte ein.



Fernschreiber F 2000 mit erweiterter Speicherkapazität

Zur Leipziger Frühjahrsmesse 1987 vom 15. bis 21. März stellt der VEB Kombinat Nachrichtenelektronik im Obergeschoß der Messehalle 15 seine nachrichtentechnischen Erzeugnisse vor. U. a. werden Lösungen auf der Grundlage von leitungsgebundenen und Funknachrichtensystemen für die Sprach-, Daten- und Textkommunikation sowie die Bewegtbild-Übermittlung vorgestellt – bestimmt für die Rationalisierung sowie Lenkung, Leitung und Überwachung von Prozessen. Im Mittelpunkt dabei steht ein Dispatcherarbeitsplatz, zu dessen Komponenten die neue universelle Vermittlungsanlage DVA 200, der weiterentwickelte Fernbildschreiber FB 2011, UKW-Verkehrsfunktechnik sowie der Personalcomputer 1715 vom VEB Kombinat Robotron zählen. Den mikroprozessorgesteuerten Sende- und Empfangsfernschreiber F 2000 für die Informationsübertragung bis 200 Baud gibt es nunmehr auch in einer Ausführung mit 32 KByte. Erweitert wurde der elektronische Textspeicher von 8 auf 26 KByte, so daß sich jetzt auch umfangreichere Texte im Lokalbetrieb aufbereiten lassen. Das Redigieren und Korrigieren des Textes ist an jeder beliebigen Stelle möglich. Automatisch zugeordnete Textnummern erleichtern die Verwaltung des Speicherinhaltes. Auf Anforderung wird ein komplettes Inhaltsverzeichnis ausgedruckt. Die gespeicherten Texte können auch mittels eines vereinbarten Schlüsselwortes vom Fernschreiber abgerufen werden.
Der Empfangsspeicher mit seiner Speicherkapazität von 6 KByte ermöglicht den ungestörten Lokalbetrieb. Alle ankommenden Nachrichten werden automatisch gespeichert und nach Erledigung der lokalen Aufgabe ausgedruckt.
Das Mikrocomputersystem gewährleistet eine wesentliche Zeiteinsparung und einfache Bedienung durch die Übernahme von Routineprozeduren beim Fernschreibverkehr wie z. B. automatisches Einfügen von Laufnummern, Uhrzeit und Datum, Kurzwahl und Wahlwiederholung oder Einfügen sich wiederholender Kurztexte (maximal 6) an jeder beliebigen Stelle der Nachricht.


Werkfoto

Dialog

Mikroprozessortechnik:
Ihre Zeitschrift für Mikroelektronik-Computertechnik-Informatik

Bereits vor Erscheinen der ersten Ausgabe von Mikroprozessortechnik (MP) erreichten uns viele Zuschriften mit Wünschen für einen guten Start und mit Hinweisen, welchen Gebieten sich die Zeitschrift zuwenden sollte. Um es gleich vorweg zu sagen, allen Vorstellungen können wir nicht gerecht werden; das, was wir uns vorgenommen haben, ist ohnehin recht anspruchsvoll: Wir wollen unsere Leser in Fachbeiträgen und Übersichtsartikeln über neue Erkenntisse und Erfahrungen in Theorie und Praxis der Computertechnik und Technischen Informatik informieren.
So werden die Entwickler und Anwender von Hard- und Software in unserer Zeitschrift u. a. Veröffentlichungen zu folgenden Gebieten finden: Schaltkreisentwurf, Schaltkreisfamilien der Computertechnik, Mikroprozessoren, Mikrocomputersysteme, Entwurfs-, Konstruktions- und Technologenarbeitsplätze, Rechnernetze, Betriebssysteme, Programmiersprachen und Softwaretechnologien.
Dieses Anliegen durchzusetzen, helfen der Redaktionsbeirat von MP, zusammengesetzt aus profilierten Vertretern von Wissenschaft und Praxis, und natürlich Sie, unsere Leser, indem Sie mit Beiträgen und Informationen von hoher Praxisrelevanz und Aktualität die Zeitschrift mitgestalten.
Mehrere Rubriken, die wir im folgenden kurz erläutern, sollen dazu beitragen, das teilweise sehr differenzierte Informationsbedürfnis zu befriedigen.
Den wachsenden Anforderungen an die Aus- und Weiterbildung soll die **Kurs**-Reihe gerecht werden. Auf vier Druckseiten je Ausgabe – in Fortsetzungen über mehrere Hefte – werden Soft- und Hardwarethemen komplex dargestellt. Dem jeweiligen Zweck entsprechend sind die Beiträge als Lehrstoff oder als Übersichtsartikel aufgebaut. Um das leichte Ausheften und Sammeln der Reihe zu ermöglichen, werden dafür die Seiten im Innenteil verwendet.
Die Nutzung bereits vorhandener Lösungen und das Vermeiden von Mehrfachentwicklungen soll die Rubrik **Börse** – siehe z. B. in Heft 1/87, S. 26 – fördern. Betriebe können in **Börse** mit maximal 15 Manuskriptzeilen Hard- und Softwarelösungen anbieten oder auch suchen.
Tagungs-, Messe- und Ausstellungsberichte findet der Leser unter der Rubrik **Bericht**. Um eine hohe Aktualität zu wahren, orientieren wir bei Tagungsberichten in der Regel auf einen Umfang von maximal 2 Manuskriptseiten.
Fachliteratur der DDR und des Auslandes wird auf der **Literatur**-Seite kritisch gewürdigt. Es sollen sowohl über interessante Veröffentlichungen in anderen Zeitschriften berichtet als auch Buchpublikationen rezensiert werden. Dabei sind die Rezensionen nicht als Werbung für die Bücher oder Broschüren zu verstehen, sondern als fachliche Einschätzung. Besonders wenn Publikationen des Auslands rezensiert werden, wollen wir damit auf internationale Entwicklungstendenzen hinweisen und – nicht zuletzt – der eigenen Buchproduktion Anregungen geben. Sollte das eine oder andere Buch aus DDR-Verlagen vergriffen sein, so nutzen Sie bitte die Ausleihmöglichkeiten der Bibliotheken.
Letztlich wäre noch unsere Rubrik **Dialog** zu nennen. Lesermeinungen, die für einen größeren Kreis von Interesse sind, werden hier veröffentlicht, ebenso wie Antworten und Mitteilungen der Redaktion. Soweit zu den Rubriken.
Im vergangenen Jahr wurden in der DDR mehr als 20 000 Personalcomputer und über 10 000 Kleincomputer ausgeliefert. So ist es nur folgerichtig, daß von der Redaktion mehrfach gefordert wurde, Mikrocomputerprogramme in MP zu veröffentlichen. Wir erfüllen diese Forderung gern, sind jedoch aus Platzgründen nicht in der Lage, umfangreiche Programme abzudrucken. Vielmehr kommt es darauf an, den besonders schöpferischen Teil eines Programms zu dokumentieren. Außerdem sollten an die Redaktion von vornherein nur Lösungen eingereicht werden, die von allgemeinem Interesse sind. Ausführliche Bedingungen für das Veröffentlichen von Programmteilen bzw. Kurzprogrammen in MP geben wir in einem späteren Heft bekannt.
Abschließend möchten wir uns für die zahlreichen Hinweise und Wünsche bedanken. Anregungen, die die inhaltliche Gestaltung von Mikroprozessortechnik betreffen, nehmen wir weiterhin gern entgegen. Ihre Redaktion MP

Hinweise für das Anfertigen von Manuskripten

1 Die mit Maschine einseitig beschriebenen Manuskripte – 2zeilig, 60 Anschläge/Zeile, 30 Zeilen/Seite – senden Sie bitte in zweifacher Ausfertigung ein. Als Manuskript können auch Computerausdrucke mit einem sauberen und kontrastreichen Schriftbild dienen, sofern die Umlaute und ß vorhanden sind.
2 Der **Umfang** des Manuskriptes soll 8 (Text-) Seiten (ausschließlich Bilder) nicht überschreiten. Auf jeden Fall muß sich der Beitrag auf maximal 3 Druckseiten in **Mikroprozessortechnik** anordnen lassen. Die maximale Länge des Textteiles von Angeboten und Suchmeldungen für die Rubrik „Börse" beträgt 15 Zeilen.
3 Das Manuskript beginnt mit einer aussagekräftigen, aber knappen Überschrift; es folgen der volle Vor- und Zuname, der akademische Grad und die Institution.
4 Bei der Gestaltung des Textteiles sollten Sie folgendes beachten:
– Beitrag übersichtlich gliedern!
(Zwischenüberschriften, Absätze; evtl. kurze Zusammenfassungen, Hervorhebung von Kernsätzen)
– Wo es der Aussage oder Anschaulichkeit dient, verwenden Sie bitte Bilder, Tafeln und Fotos (s. Pkt. 6).
5 Werden Abkürzungen verwendet, sind die im Duden aufgeführten zu nutzen; zusätzlich abzukürzende Begriffe sind beim ersten Auftreten auszuschreiben, und die Abkürzung ist in Klammern zu setzen.
6 **Bilder, Tafeln und Fotos** müssen als Anhang getrennt numeriert beigelegt werden. Die Einordnung ist im Text zu kennzeichnen. Für die erläuternden Unterschriften – sie sind bei allen Bildern, Tafeln und Fotos notwendig! – ist ein gesondertes Manuskriptblatt zu verwenden.
Bilder und Tafeln können als übersichtliche und eindeutig lesbare Bleistiftzeichnung bzw. maschinengeschrieben eingereicht werden.
Schwarzweißfotos sollten etwa das Format 13 cm × 18 cm haben (das Negativ wird nicht benötigt). Farbfotos stimmen Sie bitte mit der Redaktion ab. Geben Sie bei honorarpflichtigen Fotos den Namen und die Adresse bzw. Konto-Nr. des Bildautors an!
7 Bedenken Sie, daß die Anzahl der Zeichen pro Zeile im Druck geringer ist als im Manuskript. Passen Sie also im Interesse einer korrekten Wiedergabe Gleichungen, Formeln, Programmauszüge u. ä., die im laufenden Text erscheinen sollen, von vornherein an die **Spaltenbreite** (37 Anschläge) an (rechtzeitig trennen oder Trennstelle kennzeichnen!). Wo dieses nicht möglich ist, sind diese Zeilen als Bilder zu deklarieren.
8 **Literaturverzeichnisse** werden, in Schrägstriche eingeschlossen, fortlaufend numeriert, z. B. *wie in /1/ und /2/ eindeutig dargestellt wurde,* oder *auch Meyer hat in /3/ darauf hingewiesen.*
Das Verzeichnis der verwendeten Literatur wird auf einem gesonderten Manuskriptblatt nach folgendem Schema angefertigt:
Literatur
/1/ Menzer, R.; Richter, B.: Neue Technologie für Digitalisiergeräte. Feingerätetechnik 34 (1985) 9, S. 386
/2/ Claßen, L.; Oefler, U.: Wissensspeicher Mikrorechnerprogrammierung. VEB Verlag Technik, Berlin 1986
9 Jedem Fachartikel ist gesondert ein etwa 10 Zeilen umfassender **Referateteil** beizufügen, der das Wesentliche des Beitrages beinhaltet.
10 Bei Fachaufsätzen ist ein kurzes **Autorenporträt** – mit Angaben über Alter, Ausbildung, beruflichen Werdegang, die jetzige Tätigkeit und Aufgabenschwerpunkte – erwünscht.
11 Unterstützen Sie bitte unsere Bemühungen, den Erfahrungsaustausch zu fördern, indem Sie für Interessenten eine Kontaktadresse und Telefonnummer angeben.
12 Denken Sie daran, bei Fachbeiträgen aus Ihrem Arbeitsbereich und bei Angaben der Kontaktadresse die Veröffentlichungsgenehmigung der Dienststelle beizulegen.
13 Nicht zu vergessen sind schließlich die private und dienstliche Anschrift, Ihre Konto-Nr. sowie für evtl. Rückfragen eine Telefonnummer.
Bitte überprüfen Sie vor dem Absenden des Manuskriptes noch einmal die Einhaltung dieser Hinweise – Sie vermeiden damit unnötige Verzögerungen bei der Bearbeitung und Veröffentlichung Ihres Beitrages.

Einchip-Rechner-Schaltkreise

Architektur, Anwendung und Entwicklungsmittel

Dr. Wolfgang Fengler,
Prof. Dr. Michael Roth
Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik

1. Einleitung

Durch die Erhöhung des Integrationsgrades war es möglich, vollständige Rechnerstrukturen in einem Schaltkreis zu fertigen. Derartige Einchip-Mikrorechner-Schaltkreise (EMR) stellen in vielen Fällen die ökonomischste Variante des Einsatztes von Mikroprozessoren dar. Das resultiert daraus, daß praktisch alle Funktionen der Informationsverarbeitung durch einen Schaltkreis realisiert werden. Im Vergleich zu anderen Lösungen, die typischerweise auf Universal-Mikroprozessoren basieren, treten bei zielgerichteter Ausnutzung der Merkmale und Eigenschaften der Einchip-Rechner-Schaltkreise Verringerungen bei Preis, Volumen und Leistungsbedarf der Erzeugnisse auf. Das wird gleichzeitig von einer Erhöhung der Zuverlässigkeit begleitet, die zum großen Teil aus einer Verringerung der benötigten Verbindungen von Schaltkreisen untereinander resultiert. Im weiteren sollen Architekturmerkmale, typische Anwendungen und die für den Einsatz von EMR notwendigen Entwicklungsmittel erläutert werden.

2. Architekturmerkmale /1/.../5/

Einchip-Mikrorechner besitzen als Mindestfunktionsumfang die Zentrale Verarbeitungseinheit (CPU), den Programmspeicher, eine Form des Lese-Schreib-Speichers und digitale parallele Ein- und Ausgänge zum Peripherieanschluß. Dabei existieren Vertreter mit internem Datenformat von 4, 8 und 16 Bit. Die Leistungsfähigkeit der Zentralen Verarbeitungseinheit entspricht typischerweise CPU-Schaltkreisen im entsprechenden Datenformat. Für den Maschinen-Befehlssatz gilt das gleichermaßen. Die Tafel 1 enthält eine Übersicht über Befehlsgruppen eines typischen EMR-Vertreters der höheren Leistungsklasse.
Der Programmspeicher ist zumeist als ROM und nur in wenigen Fällen als EPROM bzw. EEPROM ausgebildet. Typische Größen sind 1 bis 16 K Byte. Der RAM-Bereich, oft in Form eines Registerfeldes ausgebildet, ist um Größenordnungen kleiner und liegt zwischen 32 und 512 Byte. Die Anzahl der digitalen Ein- und Ausgänge beträgt 16 bis 32. Leistungsfähigere Vertreter können weitere Funktionen auf dem Chip realisieren:

- *Takterzeugung* (direkter Anschluß eines Schwingquarzes oder eines Kondensators)
- *Zähler-/Zeitgeber* (mit externer und interner Triggerung, Interruptauslösung oder Ausgangssignal bei Nulldurchgang oder als Übertragungsratengenerator für den eventuellen seriellen Kanal)
- *serielle Ein- und Ausgabe* (für zumeist asynchrone Datenübertragung mit teilweiser Realisierung von Protokollfunktionen durch die Schaltkreishardware, z. B. Adreßerkennung für ein Linieninterface)
- *Quittungsbetrieb* (Handshake) (zugeordnet zur parallelen, typischerweise 8 Bit breiten digitalen Ein- bzw. Ausgabe)
- *Interruptbehandlung* (mit begrenzter Quellenanzahl; Bedingungen zur Auslösung können intern abgeleitet werden, z. B. Zähler-/Zeitgeber, serielle Schnittstelle, Quittungssignale, bzw. werden direkt bestimmten digitalen Eingängen zugeordnet)
- *Treiberstufen* (z. B. für direkten Anschluß von Anzeigen)
- *analoge Ein- und Ausgabe* (zum Teil vollständig im Schaltkreis enthalten, zum Teil als Steuerung zum Anschluß eines einfachen externen Elementes, wie Komparator, vorgesehen)
- *Kommunikationsinterface* zu einem Mikroprozessor-Systembus (über Ein-/Ausgabe-Register oder Bereiche; die Kommunikation erfolgt extern über den üblichen Buszyklus wie Ein-/Ausgabe-Zugriff; sowohl extern als auch intern kann der Datenaustausch über Interrupt gesteuert werden)
- *Bildung von Adreß-, Steuer- und Datenbus* für den externen Speicher- und E/A-Anschluß (zur Erweiterung des internen Programm- und Datenbereichs und der im Schaltkreis integrierten Ein-/Ausgabe-Funktionen, maximaler Bereich typisch 64 K Byte).

Dabei existieren nicht alle Funktionen gleichzeitig, und sie reduzieren i. a. die Anzahl der parallelen E/A-Anschlüsse, deren Funktion programmtechnisch geändert werden kann.
Bild 1 zeigt ein Funktions-Blockschaltbild eines typischen Vertreters der höheren Leistungsklasse.
Tafel 2 enthält eine Übersicht zur Architektur-Entwicklung von Einchip-Mikrorechnerschaltkreisen auf Basis ausgewählter Typen.

3. Anwendung /6/.../10/

Die erste Hauptanwendungsgruppe ist die der Geräterechner. Dabei übernimmt der EMR praktisch alle Informationsverarbeitungsfunktionen eines Gerätes, wie Meßwerterfassung, Steuerung und Bedienerkommunikation. Diese

Tafel 1 Befehlsgruppen des EMR U 881x /5/ als Übersicht

Befehlsgruppe	Untergruppen
Ladebefehle	Umladen von Registern Laden von Registern mit externem Speicherinhalt und umgekehrt Kellerspeicherzugriff Blocktransfer auf dem externen Speicherbereich
Arithmetik-/Logikbefehle	Addition, Subtraktion, Vergleich Und, Oder, Exklusiv-Oder, Test mit Maske Komplement Dezimalkorrektur Inkrementieren, Dekrementieren Rotation und Verschieben
Programmverzweigende Befehle	Sprungbefehle Unterprogrammruf und -rückkehr
Steuerbefehle	Flag- und Interruptbefehle




Dr.-Ing. Wolfgang Fengler studierte an der TH Ilmenau von 1970 bis 1974; Diplomabschluß auf dem Fachgebiet Technische Kybernetik. Anschließlich war er in der Technischen Direktion des Kombinates Carl Zeiss JENA mit dem Aufgabengebiet Fertigungssteuerung beschäftigt, von 1976 bis 1978 im VEB Kernkraftwerk Greifswald als Gruppenleiter im Bereich Wartung und Einsatzvorbereitung für Prozeßrechentechnik tätig, von 1979 bis 1983 als wissenschaftlicher Mitarbeiter im VEB Wägetechnik Radebeul. Während dieser Zeit Promotion auf dem Gebiet der Hard- und Software-Entwurfsmethodik von Geräterechnern, seitdem wissenschaftlicher Oberassistent und Forschungsgruppenleiter des Bereichs Computertechnik an der TH Ilmenau.
Prof. Dr. sc. phil. Dr.-Ing. Michael Roth diplomierte 1963 auf dem Gebiet der Regelungs- und Steuerungstechnik an der TH Ilmenau; 1967 Promotion A zum Hybridrechnerentwurf; 1967/1968 Zusatzstudium am Moskauer Energetischen Institut. 1969 erfolgte die Berufung zum Hochschuldozenten und 1978 die Berufung zum Ordentlichen Professor. Prof. Roth ist Leiter des Wissenschaftsbereiches Computertechnik, Sektion Technische und Biomedizinische Kybernetik an der Technischen Hochschule Ilmenau.

Tafel 2 Entwicklungstendenzen von EMR-Architekturen /21/

Schaltkreisbezeichnung	Herstellerfirma	Erscheinungsjahr	Einsatzgebiet, charakteristische Eigenschaften	Literaturangabe
8048 (SAB 8048)	Intel (Siemens)	1977	universeller EMR zum Einsatz in Haushaltgeräten, Kfz, Kassen, Terminals; 64 Byte int. RAM, 1 K int. ROM, 24 dig. E/A-Leitungen, Zähler/Zeitgeber	/1/
SAB 80C482	Siemens	1982	spezielle Version des 8048 für Telefonanwendungen	/17/
8051 (SAB 8051)	Intel (Siemens)	1982	Weiterentwicklung des 8048 mit erweiterten E/A-Möglichkeiten und Befehlsvorrat, leistungsfähigere serielle Schnittstelle mit Multimasterfähigkeit (I^2C)	/17/
80515	Intel	1984	speziell für Steuerungs- und Regelungsaufgaben weiterentwickelter 8051, z. Z. leistungsfähigstes Mitglied der 8015-Familie, intern 8 K Programmspeicher, 256 Byte interner Datenspeicher, 56 E/A-Anschlüsse, 8 Analogeingänge, 3 16-Bit-Zähler/Zeitgeber, Watchdog-Timer, ausgefeilte Bitmanipulationsbefehle	/18/
MAB 8409	Phillips	1983	EMR-Familie, 1–4 KByte ROM, 32–128 Byte RAM, 22 E/A-Leitungen serielles I^2C-Bus-Interface	/16/
Z8601/ Z8611	Zilog	1978/ 1980	universeller EMR der höheren Leistungsklasse, 2 K bzw. 4 K interner Programmspeicher, 32 E/A-Leitungen, 1 serielle Schnittstelle, 2 Zähler/Zeitgeber	/3/
Z8034 UPC	Zilog	1980	universeller Peripheriecontroller, basiert auf Z8-Architektur, voller Z8-Befehlssatz, Interface für Z-Bus zur Kopplung mit Masterrechner, 256 Byte Registerfile	/3/
M6805	Motorola		EMR-Familie für Steuerungsanwendungen, Kernstück ist CPU, die für alle gleich ist. Spezielle Versionen beinhalten folgende zusätzliche Funktionen: A/D-Wandlung, Zähler/Zeitgeber, Ansteuerung von TTL- und CMOS-Lasten, LEDs	/19/
3870 (Basisrechner)	Fairchild		„dedicated" Microcomputer; Angebot eines Chips, das als Basis einen 3870 enthält und auf das der Kunde weitere Schaltkreise integrieren lassen kann (z. B. A/D-, D/A-Wandler, Leistungsein- und -ausgänge)	/17/
TMS 32010	Texas Instruments	1983	besonders geeignet für rechentechnisch intensive Aufgaben (z. B. Filter und Sprachein-/Ausgabe-Systeme) 32-Bit-Struktur, 5 Mill. Instruktionen/Multiplikationen in einem Taktzyklus ausführbar, serielle Schnittstelle	/17/
TMS 32020	Texas Instruments	1984	Weiterentwicklung des TMS 32010, parallele Schnittstelle für Multiprozessorsysteme	/20/

Aufgaben, die bisher durch kleine modulare Systeme oder Einkartenrechner realisiert wurden, gewinnen durch den EMR eine völlig neue ökonomische Dimension, die diese Art der Informationsverarbeitung bis in den Bereich von Konsumgütern vordringen läßt. Bild 2 zeigt eine Konfiguration aus diesem Anwendungsgebiet. Charakteristisch ist, daß weitgehend auf Zusatzschaltkreise verzichtet wird und die Verlagerung der meisten Hardware-Funktionen in die Software erfolgt. Der Einchip-Rechner dient zur Qualifizierung eines Sensors mit analogem Meßwertausgang. Der angegebene Geräterechner übernimmt die Analog-Digital-Wandlung der Eingabegröße (ADU), die Steuerung des Meßablaufs (Binär-EA), die Bediener-Kommunikation und Meßwertanzeige (Tastatur/Anzeige), die Bereitstellung maschinell verarbeitbarer Ausgangsdaten (Binär-EA) und die Primär- und Sekundärverarbeitung der Meßgröße (EMR).

Dazu können gehören:
– Störsignalunterdrückung
– statistische Aufbereitung (z. B. Mittelwertbildung) und
– Erzeugung abgeleiteter Größen (z. B. Umrechnung eines gemessenen Gewichtes in einen Preis).

Einchip-Rechner mit seriellem Interface bilden die Basis hochleistungsfähiger, dezentraler Modulsysteme der Prozeßdatenverarbeitung (Bild 3). Durch Zuordnung eines EMR zu einer oder wenigen Meß- und Stell-Einrichtungen entsteht der Vorteil einer großen parallel arbeitenden Verarbeitungskapazität und einer hohen Systemzuverlässigkeit bei geringem Verkabelungsaufwand für die gesicherte Datenübertragung.
Als weiterer Vorteil kann die Software weitgehend objektorientiert und unabhängig für die einzelnen Prozesse entworfen werden. Der Vorfertigungsgrad der Programme kann hoch sein.
Die Tafel 3 zeigt eine Übersicht über ein derartiges Modulsystem.
Entsprechende Vorteile besitzt der Einsatz von EMR als universelle Peripherie-Controller (UPC). Diese entlasten eine System-CPU von peripheren Steueraufgaben bei weitgehender Flexi-

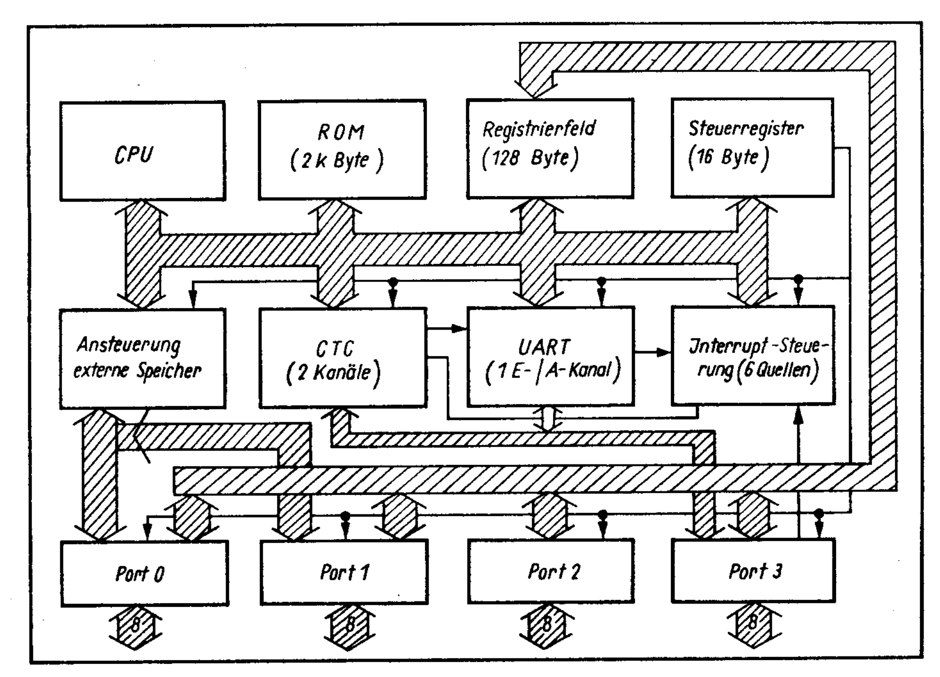


Bild 1 Funktions-Blockschaltbild eines typischen EMR

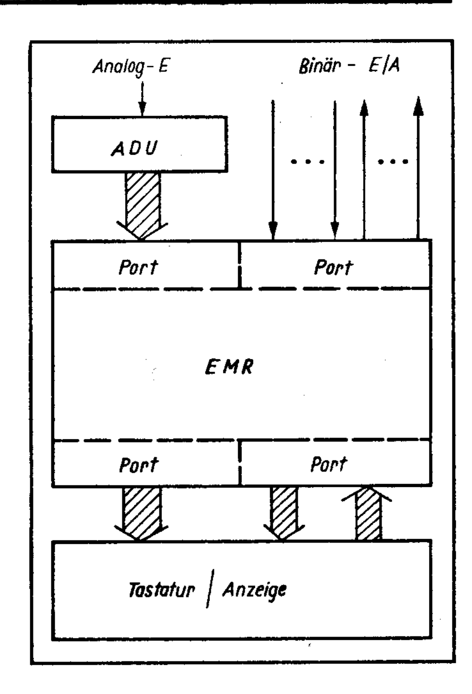


Bild 2 EMR als Geräterechner

bilität für den Peripherie-Anschluß. Notwendig ist ein Interface des EMR zum Systembus. Aus diesem Grund werden spezielle UPC-Versionen von EMR angeboten, die teilweise auch mit ROM-Masken, wie Druckersteuerung oder GPIB-Interface-Controller, ausgerüstet sind. Bild 4 zeigt die Systemeinordnung eines UPC.
Ein weiterer Einsatzschwerpunkt ist die Bildung einer hochsprachfähigen Makro-CPU. Dabei wird der EMR mit externem Speicheranschluß genutzt und bildet einen normalen Mikroprozessor-Systembus. Der interne ROM enthält einen Interpreter für eine geeignete Sprache, wie z. B. BASIC oder FORTH, die dadurch die Nutzerebene des Anwenders bildet.

Bei der Implementierung eines FORTH-Systems wird für den Einsatz als Spezialmikrorechner, wie er beim Geräte-Rechner bzw. beim modularen Prozeßdatenverarbeitungssystem auftritt, nur der innere Interpreter im EMR realisiert. Dafür reicht der im ROM enthaltene Speicherumfang aus.
Aufgrund der sprachtypischen Fadenkode-Interpretation liegt die Abarbeitungsgeschwindigkeit in vertretbaren Größenordnungen, der Programmkode (reduziertes Wörterbuch) ist relativ kurz. Dabei kommen alle wesentlichen Vorteile des Einsatzes einer höheren Programmiersprache, wie Portabilität, Möglichkeit der strukturierten Programmierung und höheres Entwurfsniveau, zum Tragen. Darüber hinaus ist der Einsatz von FORTH für die Implementierung von Fachsprachen aufgrund des speziellen Wortkonzeptes besonders geeignet. Sinnvoll ist eine Erweiterung um im Standardumfang nicht enthaltene Echtzeit-, Ein-/Ausgabe- und Gleitkomma-Arithmetik-Funktionen auf dem Sprachnivau. Eine Übersicht für eine FORTH-Implementierung auf dem EMR U881x bzw. U882x enthält Tafel 4.

4. Entwicklungsmittel /11/.../13/

Für den Programmentwurf für Einchiprechner stehen die üblichen Hilfsmittel wie Editoren, Assembler, Binder und teilweise Compiler zur Verfügung. Da die Programmierung von EMR aufgrund der begrenzten ROM-Größe z. Z.

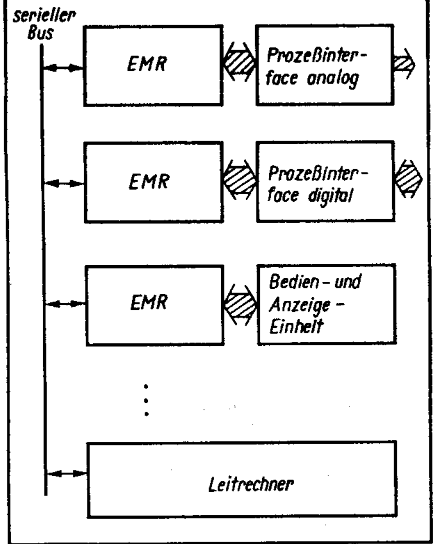


Bild 3 EMR-Einsatz im Modulsystem zur Prozeßdatenverarbeitung

Tafel 3 Übersicht über das module Mikrorechnersystem MMS 881 /14/

Merkmal	Realisierung im MMS 881
Systembus	Serieller Multi-Master-Bus, Physische Ebene: 20-mA-Stromschleife, asynchron, 62,5 KBaud, 15 Teilnehmer an einer Linie. Logische Ebene: Token-Ring mit zyklischer Masterweitergabe und verschiedenen Sicherungsmaßnahmen
Modultypen	Analoge Ein- und Ausgabe-Module Regler-Module, Steuerungsmodule Inkremental- und Frequenzeingänge Kommunikationsmodule Leitgeräte Verarbeitungsmodule für allgemeine Aufgaben Standard-Interface-Module Module zur Leitrechner-Kopplung
Firmware	Regelungstechnische Fachsprache Steuerungstechnische Fachsprache Kommunikations-Fachsprache Vorgefertigte Interface-Programme Echtzeit-FORTH
Einsatzgebiet	Dezentrale Prozeßdatenverarbeitung (Klein- und Mittelautomatisierung)

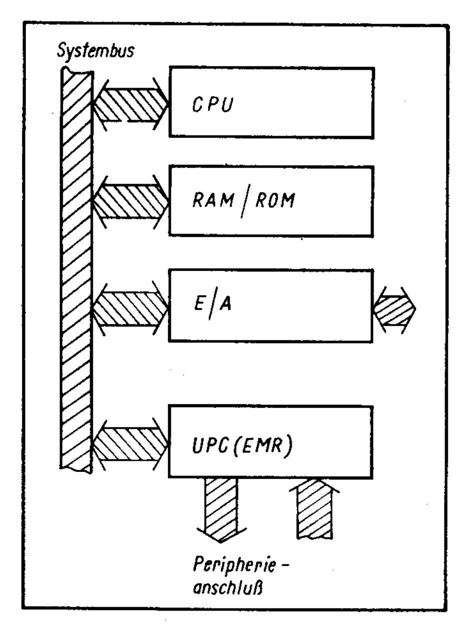


Bild 4 EMR als universeller Peripherie-Controller (UPC)

Bild 5 Entwicklungschip eines EMR

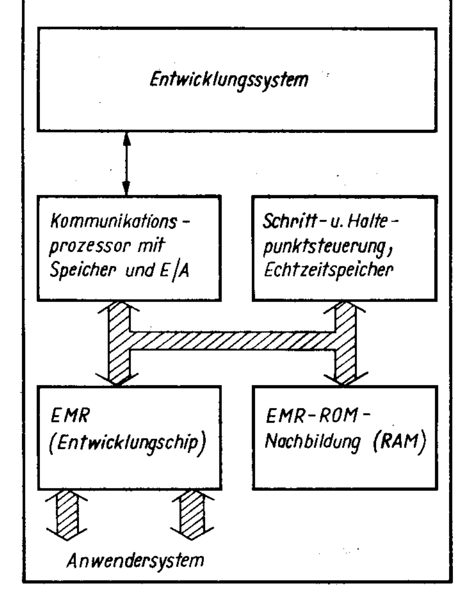


Bild 6 Emulationsmodul für EMR-Hard- und Softwareentwicklung

vorwiegend in Assembler erfolgt, sollten hier zumindest strukturierte Assemblersprachen zur Verfügung stehen.
Der Einsatz höherer Sprachen ist bei EMR am effektivsten mit im ROM enthaltenen Firmware-Interpretern. Dabei stellt sich die Sprache FORTH aufgrund der hohen Abarbeitungsgeschwindigkeit, der relativen Maschinennähe und der Flexibilität als besonders geeignet dar.
Für den Programmtest benötigt man auf Entwicklungssystemen lauffähige Simulatoren, die allerdings keinen Echtzeittest ermöglichen. Dafür sind EMR mit integriertem EPROM oder EEPROM oder spezielle Entwicklungschips mit extern anschließbarem EPROM notwendig. Bild 5 zeigt einen derartigen Schaltkreis. Das Anwenderinterface entspricht den Versionen mit internem ROM. Die effektivste Nutzung dieser Entwicklungs-EMR erfolgt in Emulatoren, die von einem Entwicklungssystem bedient werden, in dem der Programmentwurf realisiert wird. Zusätzlich zum Entwicklungschip enthält der Emulator eine ROM-Nachbildung als ladbaren RAM, Steuerungen für Schrittbetrieb und Haltepunkte, einen Echtzeitspeicher zur Analyse der Vorgeschichte bei Erreichen eines Haltepunktes und einen Kommunikationsprozessor (Bild 6). Letzterer steuert den gesamten Emulationsverlauf und bildet das Interface zum Entwicklungssystem. Dabei wird er im allgemeinen selbst durch einen EMR realisiert.
Eine Kommando-Übersicht über den minimalen Funktionsumfang eines In-circuit-Emulators für die EMR-Familie U881x und folgende enthält Tafel 5.

Literatur

/1/ Eigenschaften und Anwendung von Einchip-Mikrorechnern (Literaturbericht). Wissenschaftliche Informationen und Berichte 10, VEB Kombinat Robotron 1979
/2/ Intel Component Data Catalog 1980. Santa Clara, CA: Intel Corporation 1980
/3/ Zilog 1983/84 Data-Book. Cupertino, CA: Zilog Inc. 1983

Tafel 4 Übersicht zu einer FORTH-Implementierung für Verarbeitungsmodule im MMS 881 /7/, /14/

Befehlsgruppe	Realisierung im FORTH8 des MMS 881
Standardwortschatz	Festkomma-Arithmetik-Befehle Programmsteuernde Befehle (für strukturierte Programmierung) Daten- und Returnstack-Befehle Definierende Befehle Speichermanipulation Logische Befehle
Gleitkomma-Erweiterung	Arithmetik-Befehle
Echtzeit-Erweiterung	Multi-Task-Befehle Tasksynchronisation Echtzeit-Uhr-Funktionen
Kommunikations-Erweiterung	Synchronisierter Zugriff auf den Multi-Master-Systembus auf Befehlsniveau mit Tasksynchronisation

Tafel 5 Funktionsumfang eines In-circuit-Emulators für die EMR-Familie U881x und folgende /15/

Kommando	Bedeutung
N	Ausführung des nächsten Maschinenbefehls
B adr	Haltepunkt auf Adresse adr setzen
K adr	Haltepunkt auf Adresse adr löschen
K	alle Haltepunkte löschen
P (adr') adr	Anzeige und eventuelle Manipulation des Befehlszählers
G	Weiterarbeit ab (eventuell über P manipuliertem) aktuellem Befehlszähler
M	Unterbrechung des laufenden Programms
R adr	Anzeige und eventuelle Manipulation des Registers adr
D adr	Anzeige und eventuelle Manipulation des Speicherplatzes adr
GE	Laden einer Datei in den Programmspeicher
SA	Auslagern des Programmspeicherinhaltes auf eine Datei
I E/D	Zulassen bzw. Sperren von Interrupts

/4/ Periperal Design Handbook. Santa Clara: CA: Intel Corporation 1982
/5/ Bankel, M.: Einchip-Mikrorechner U881, U882 und U883. rfe 34 (1985) 2, S. 81–84
/6/ Fritz, K.-D.; Geidel, K.-D.: Einkanalregler mit Einchip-Mikroprozessor. rfe 30 (1981) 11, S. 723
/7/ Fengler, W.: FORTH-Interpreter für den Einchip-Mikrorechner Z8. 1. Fachtagung „Die Programmiersprache FORTH – Grundlagen, Möglichkeiten, Anwendungen", Leipzig: 29.3.1984
/8/ Multiprozessor-Systeme mit Single-Chip-Mikrocomputern. Elektronik-Applikation 15 (1983) 4, S. 33–43
/9/ Pelka, H.: Der Einchip-Mikrocomputer. Franzis-Verlag, München 1981
/10/ Schwartz, F.: Einsatzmöglichkeiten des Einchip-Mikrorechners U881/U882 in prozeßnahen Automatisierungseinrichtungen. msr 28 (1985) 3
/11/ Handbuch UDOS-1526. Sprachbeschreibung /Manual, Crossoftware U 881/882 ASM. Robotron Anwenderdokumentation, VEB Buchungsmaschinenwerk 1984
/12/ Z8 Development Module User Manual. Cupertino, CA: Zilog Inc. 1980
/13/ Geidel, K.-D.; Kademowa-Katzarowa, P.; Roth, M.: Universelle modulare Emulatoren. rfe 33 (1984) 12, S. 761–762
/14/ Fengler, W.: Modulares System der dezentralen Prozeßdatenverarbeitung. KDT-Fachtagung „Computer- und Mikroprozessortechnik '86", Berlin: 28.–29.10.1986
/15/ Fengler, W.; Kotula, H.: Bedienungsanleitung zum U881-In-circuit-Emulator. Dokumentation zur Forschungsnachnutzung, Ilmenau: Technische Hochschule 1986
/16/ Keve, E. T.: Circuits for modular design of consumer and industrial products – the CLIPS system. Electronic components & applications 1983/2, S. 29 ff.
/17/ Mikroprozessoren und Einchip-Computer. Elektronik-Applikation 14 (1982) 12, S. 56–61
/18/ Feger, O.; Stärk, J.; Storandt, S.: Einchip-Mikrocomputer für Steuerungs- und Regelungsaufgaben. Elektronik 1984/21, S. 67–72
/19/ Michnal, J.; Farell, J. J.: Einchipcomputer für Steuerungsanwendungen konzipiert. Elektronik 1984/6, S. 110–114
/20/ Kropp, C.: Signalprozessor bietet Multiprozessorfähigkeit. Elektronik 1985/6
/21/ Schlott, St.: Softwarearchitektur eines modularen Mehrrechnersystems der dezentralen Prozeßdatenverarbeitung. Dissertation A, Ilmenau: Technische Hochschule 1986

KONTAKT

Technische Hochschule Ilmenau,
Sektion Technische und Biomedizinische Kybernetik,
Wissenschaftsbereich Computertechnik,
PSF 327, Ilmenau, 6300; Tel. 74509,
Dr. Fengler,

Meßwerterfassung mit CCD-Sensoren

Dr. Bernd Michaelis, Rainer Maaß
Technische Hochschule Magdeburg,
Sektion Technische Kybernetik
und Elektrotechnik

1. Einleitung

Die Informationsgewinnung mit CCD-Zeilen- und CCD-Matrixsensoren hat in den letzten Jahren eine große Bedeutung erlangt /1/.../4/. Da Zeilen- und Matrixsensoren zum Teil aus einigen Tausend einzelnen Sensorelementen bestehen, liefert ein Meßvorgang im Vergleich zu herkömmlichen Systemen eine sehr viel höhere Informationsdichte. Darin zeigt sich einerseits die neue Qualität, andererseits werden aber auch die Anforderungen an die Verarbeitungskapazität deutlich. Es ist leicht einzusehen, daß die Vorteile dieser Technik nur in Verbindung mit einer leistungsfähigen Mikrorechentechnik voll genutzt werden können.
Bei voller Ausschöpfung der angebotenen Information (Bildverarbeitung) und hoher Meßdynamik entsteht jedoch durch die in der Regel sehr hohen Anforderungen an die rechentechnische Verarbeitungsgeschwindigkeit ein ernstes Problem /5/. Es kann der Einsatz leistungsfähiger Rechner bzw. von Spezialprozessoren erforderlich sein.
Auch die einfache Bestimmung geometrischer Abmessungen (Länge, Durchmesser u. ä.) mit Zeilensensoren erfordert bei programmtechnischer Auswertung des entstandenen Bildes /6/, /7/ Zeitintervalle, die für kontinuierliche Messungen, beispielsweise im Walzprozeß, oft schon viel zu groß sind. Deshalb ist es zweckmäßig, in Analogie zu speziellen Prozessoren bei der Bildverarbeitung, auch bei der Gewinnung einfacher Meßgrößen aus dem vorliegenden Informationsangebot eine spezielle Vorverarbeitungselektronik vorzusehen, um die erforderliche Meßdynamik zu gewährleisten. Auf diese Problemstellung soll nachstehend näher eingegangen werden.

2. Struktur der Meßwerterfassungseinrichtung

In Meßwerterfassungseinrichtungen auf der Basis von CCD-Sensoren wird das Meßobjekt mit Hilfe eines Objektivs möglichst maßstabgerecht, unverzerrt und scharf auf den Sensor abgebildet. Das Videosignal (Ausgangssignal des CCD-Sensors) ist ein serielles Analogsignal, das alle weiterzuverarbeitenden Informationen enthält. Daraus ist die gesuchte Meßgröße (oder mehrere) zu bestimmen. Bei genügend langsamen Vorgängen kann dies nach Binarisierung bzw. Analog-/Digital-Umsetzung und Zwischenspeicherung der den einzelnen Sensorelementen zugeordneten Werte durch Abarbeitung geeigneter Algorithmen erfolgen. Läßt sich die notwendige Meßdynamik so nicht realisieren, bildet das Einfügen einer auf das konkrete Problem zugeschnittenen Vorverarbeitungselektronik einen Ausweg. Man erhält das in Bild 1 gezeigte Blockschaltbild. Auf Videoverstärker und Takterzeugung soll in diesem Zusammenhang nicht näher eingegangen werden. Sie sind entsprechend den eingesetzten Sensoren und der benötigten Grenzfrequenz auszulegen.
Für die Vorverarbeitungselektronik lassen sich nur schwer allgemeine Aussagen treffen. Im wesentlichen geht es darum, den nachgeschalteten Mikrorechner von zeitaufwendigen Operationen zu entlasten, indem dafür eine spezielle Elektronik bereitgestellt wird. Im einfachen Fall der Längenmessung können dies Komparatoren zur Ermittlung der Kante im Videosignal und geeignete Zähler zur numerischen Bestimmung der Lage der beiden Kanten bzw. ihres Abstandes sein (siehe auch Bild 2). Auf diese Weise lassen sich bei eingeschränkten Anforderungen auch prozessorlose Meßgeräte zur Durchmessermessung aufbauen /8/.
Stellt man die Forderungen etwas höher, so steigt der Aufwand schnell an. Schon bei der Auswertung des Kantenverlaufs in Bild 2 ist eine A/D-Umsetzung und arithmetische Verarbeitung der Werte in der Umgebung der Kante erforderlich. Bei der Messung von Bewegungsgeschwindigkeiten, der Bewer-

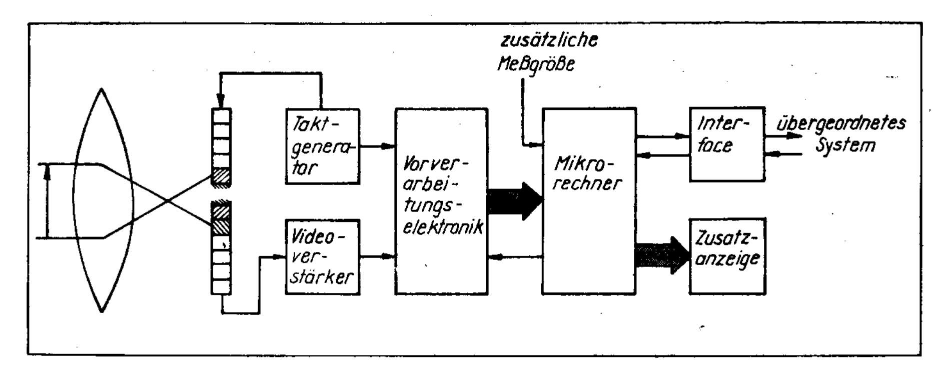


Bild 1 Allgemeines Blockschaltbild der Meßwerterfassungseinrichtung



Doz. Dr. sc. techn. Bernd Michaelis (39) studierte von 1966 bis 1971 an der Technischen Hochschule Magdeburg, Sektion Technische Kybernetik und Elektrotechnik. Im Anschluß an ein Forschungsstudium an der gleichen Sektion erfolgte im Jahre 1974 die Verteidigung der Dissertation A. Ab 1973 wissenschaftlicher Assistent im Wissenschaftsbereich Prozeßtechnik und 1980 Verteidigung der Dissertation B. Von 1980 bis 1983 Delegierung an das VIK-Dubna (UdSSR) für Arbeiten auf dem Gebiet des Mikrorechnereinsatzes. 1984 Berufung zum Hochschuldozenten an die Technische Hochschule Magdeburg. Derzeitige Forschungsarbeiten betreffen Meßwerterfassung mit optoelektronischen Sensoren und Signalauswertung mit Mikrorechnern.

Dipl.-Ing. Rainer Maaß (26) studierte von 1980 bis 1985 an der Technischen Hochschule Magdeburg, Sektion Technische Kybernetik und Elektrotechnik. Seit 1985 arbeitet er als wissenschaftlicher Assistent im Wissenschaftsbereich Prozeßmeßtechnik dieser Sektion und beschäftigt sich mit Forschungsarbeiten auf dem Gebiet der Meßwerterfassung mit optoelektronischen Sensoren und Signalauswertung mit Mikrorechnern.

tung von Oberflächenstrukturen usw. gilt häufig ähnliches. Es gibt einen kontinuierlichen Übergang bis zu Erkennungsproblemen bei der Robotersensorik, wo die hier angerissenen Probleme eigentlich selbstverständlich sind. An dieser Stelle soll auch auf die in bestimmten Fällen im optischen Bereich mögliche Vorverarbeitung hingewiesen werden.
Für die Vorverarbeitungselektronik bieten sich je nach Leistungsfähigkeit grob folgende Varianten an:

- Einfache Schaltung auf der Basis von Standardschaltkreisen
- aufwendige Spezialelektronik
- spezielle Schaltkreise auf der Basis der Gate-Array-Technik u. ä.
- Prozessoren für spezielle Verarbeitungsaufgaben (U882, U8000 u. ä.) mit Zusatzelektronik.

Auf ein Realisierungsbeispiel für die zuerst genannte Variante wird im dritten Abschnitt eingegangen (siehe auch /9/).
An die Vorverarbeitungselektronik schließt sich in Bild 1 ein Mikrorechner an. Er ermittelt die gesuchte Meßgröße aus den Ergebnissen der Vorverarbeitungselektronik und steuert die Kommunikation mit dem übergeordneten Rechner bzw. unmittelbar die Anzeige. Weiterhin können auch (langsam variable) Einstellparameter für die Vorverarbeitungselektronik bestimmt werden. Bei praktischen Messungen ist es teilweise erforderlich, die mit Hilfe der optoelektronischen Messung gewonnenen Größen mit weiteren Meßgrößen (z. B. Längenimpulse von bewegtem Gut) zu verknüpfen, um das endgültige Resultat zu erhalten /10/.
Gleichfalls muß, gesteuert durch diesen Mikrorechner, im Interesse der Betriebssicherheit des Systems eine (zumindest teilweise) Selbstüberwachung erfolgen /4/.

3. Realisierungsbeispiel

Ein einfaches Beispiel für das vorgestellte Konzept stellt ein intelligentes Durchmessermeßgerät /9/ dar. Es enthält als Verarbeitungselektronik hauptsächlich einfache Zähler, einen Komparator zur Binarisierung u. ä. Diese Baugruppe kann bei erweiterten Meßaufgaben aufgrund des modularen Aufbaus durch eine komfortablere ausgetauscht werden, ohne daß dadurch eine Neukonzeption des gesamten Meßsystems erforderlich wird. Darauf soll in späteren Arbeiten eingegangen werden.
Bild 3 zeigt das Blockschaltbild des realisierten intelligenten Durchmessermeßgerätes auf der Basis der Einchipmikrorechners U882 und des Zeilensensors L 133 C.
Ein Objektiv bildet den Prüfling (z. B. Draht) auf der Sensorzeile ab. Entsprechend der Beleuchtungsstärke baut sich in den einzelnen Sensorelementen der Zeile ein Spannungssignal auf. Im Rhythmus der verwendeten Taktfrequenz wird die in den Sensorelementen gespeicherte Information herausgeschoben und steht als serielles Analogsignal zur Verfügung, das die Helligkeitsverteilung über die Zeile wiedergibt. Bild 2 zeigt den typischen Verlauf eines solchen (verstärkten) Videosignals U_{Video}. Zur rechnerischen Korrektur von optischen Verzerrungen ist es zweckmäßig, nicht nur diesen Abstand, sondern die Lage beider Kanten zu kennen, die durch l_1 bzw. l_2 gekennzeichnet sind. Ein Komparator mit geeigneter Vergleichsspannung /1/ erzeugt aus dem analogen Videosignal eine rechteckförmige Spannung, die zur Steuerung von Zähltoren dient. Zwei Zähler mit je 12 Bit Breite bestimmen dann die Lage der Kanten über die Nummer des zugeordneten Sensorelementes (Pixel). Pufferregister für die verwendeten Zähler sichern einen gewissen Spielraum für den konkre-

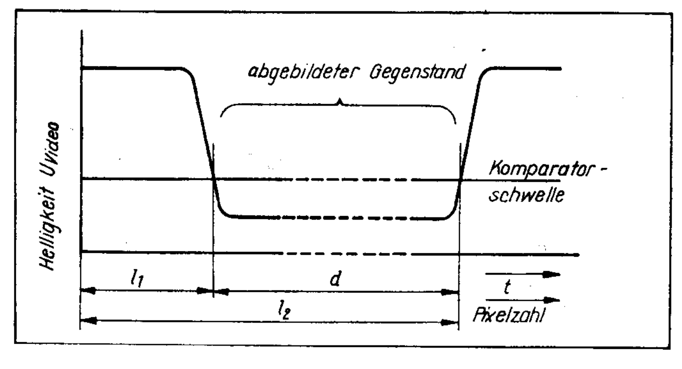


Bild 2 Prinzipieller Verlauf des Videosignals bei der Durchmesserbestimmung

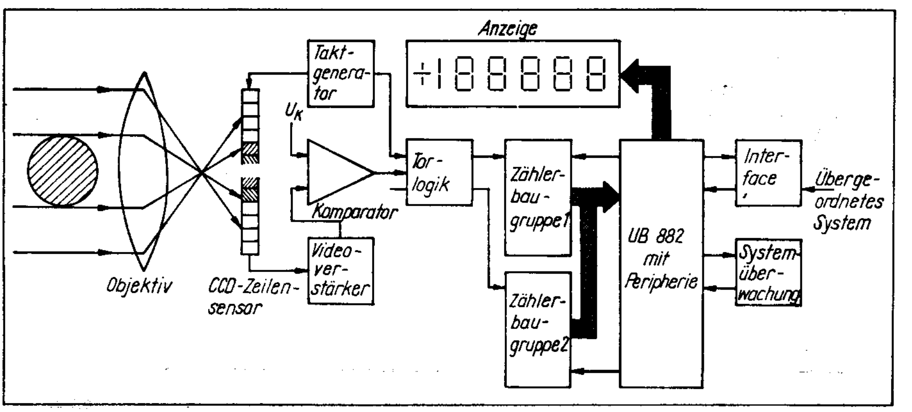


Bild 3 Intelligentes Durchmessermeßgerät

ten Zeitpunkt des Einlesens der Meßwerte. Der Rechner U882 erhält diese Information über Interrupt bzw. durch das Setzen eines Statusbits.
Ergänzend zum internen Programmspeicherbereich des U882 erfolgte eine Erweiterung, um bei der Programmierung genügend flexibel zu sein. Da die internen Zeitgeber für den gesamten Aufgabenkomplex nicht ausreichen, wurde das System durch einen CTC-Baustein aus dem U-880-System erweitert. Damit kann auch die Integrationszeit T_{int} der Zeile vom Rechner programmiert werden. Des weiteren stehen zusätzliche Zähler zur Überwachung der Programmlaufzeit zur Verfügung. Trotz des Einsatzes peripherer Bausteine neben dem U882 ergaben sich im Vergleich zu einer Lösung mit dem Mikroprozessor U880 D für den vorliegenden Anwendungsfall durch die vielen integrierten Funktionen bei relativ leistungsfähigem Befehlssatz Vorteile bezüglich Platzbedarf und Leistungsverbrauch.
Die integrierte UART wird für die Kommunikation mit dem übergeordneten System über eine serielle Standardschnittstelle genutzt. Ein Port des Einchiprechners steuert die unmittelbare Anzeige der Meßergebnisse am Gerät über Lichtschachtbauelemente. Weiterhin sind für das Interface zur Vorverarbeitungselektronik einige Ein- bzw. Ausgabeleitungen erforderlich.
Die verschiedenen Verarbeitungs- und Überwachungsfunktionen werden durch priorisierte Interruptroutinen realisiert. Es handelt sich hauptsächlich um
- Programmlaufzeitüberwachung
- Reaktion auf Meßbereichsüberschreitungen
- Meßwerteingabe, Durchmesserberechnungen und exponentielle Mittelwertbildung der Meßwerte
- serielle Schnittstelle (IFSS)
- arithmetische Operationen und Anzeige der Meßergebnisse
- (bei Bedarf:) zusätzliche Rechenoperationen zur Ermittlung des Meßergebnisses (z. B. Korrektur optischer Verzerrungen, Verknüpfungen mit weiteren Meßgrößen), Ansteuerung von Zusatzgeräten.

Dabei sind die Prioritäten so gestaffelt, daß im fehlerfreien Betrieb alle von der CCD-Zeile zyklisch bereitgestellten Meßwerte ausgewertet werden. Durch die gleitende (expotentielle) Mittelung der errechneten Meßwerte gelingt es, den Einfluß statistisch verteilter Störungen wesentlich zu vermindern und trotzdem eine gute Dynamik sicherzustellen. Wird die Zeile im Bereich einiger Megahertz betrieben (im Testmuster liegt softwarebedingt die erreichbare Zeilenauslesefrequenz bei etwa 4 MHz), ist damit die Leistungsgrenze des Prozessors für eine on-line-Verarbeitung nahezu erreicht. Die verbleibende Rechenzeit wird für die Anzeige von Ergebnissen, die Kommunikation mit dem übergeordneten Rechner u. ä. sowie das Hintergrundprogramm genutzt. Einige eingebundene 3-Byte-Gleitkommaarithmetik ermöglicht die Kompensation veränderlicher Offsetgrößen und die Skalierung der Meßergebnisse.
Es wird deutlich, daß trotz getrennter Vorverarbeitungselektronik, die bereits die Lage der Kanten des Meßobjekts bestimmt, die Bereitstellung einiger Tausend Durchmesserwerte pro Sekunde kaum zu überschreiten ist. Bei schnell beweglichem Meßgut (z. B. Walzvorgänge) wird diese Größenordnung durchaus benötigt, auch wenn es oft nicht erforderlich ist, jeden einzelnen Wert an das übergeordnete System weiterzugeben, sondern nur vorverdichtete Größen.
Schon in dieser einfachen Form steht damit ein „intelligentes" Meßgerät zur Verfügung, das durch Modifikation und Erweiterung der Software für eine Vielzahl industrieller Meßprobleme geeignet ist, die sich auf Durchmesserbestimmungen zurückführen lassen.

4. Schlußfolgerungen

Die Ausführungen zeigen, daß bei hohen Anforderungen an die Meßdynamik schon relativ einfache Meßaufgaben mit optoelektronischer Informationserfassung es zweckmäßig erscheinen lassen, nicht das gesamte Bild rechentechnisch auszuwerten, sondern durch eine spezielle Vorverarbeitungselektronik problemangepaßte Größen zu gewinnen, aus denen dann mit Hilfe eines Mikrorechners die gesuchte Meßgröße ermittelt und an das übergeordnete System weitergegeben wird. Dieser Mikrorechner kann zusätzlich Mittelwertbildungen, Skalierungen u. ä. sowie das Übertragungsprotokoll realisieren. Die Vorverarbeitungselektronik entlastet den Mikrorechner und muß je nach Aufgabenstellung selbst aus leistungsfähigen Prozessoren aufgebaut werden.

Literatur

/1/ CCD-Bildsensortechnik. Technische Hochschule Ilmenau/KDT Bezirksvorstand Suhl 1985
/2/ Chromov, L. I. u. a.: Tvordotel'noe televidenie (Festkörperfernsehen, russ.). Moskau 1986
/3/ Bleicher, M.: Halbleiter-Optoelektronik. Berlin 1986
/4/ Heimburger, T.; Michaelis, B.; Wartini, Chr.: Einsatzmöglichkeiten optoelektronischer Sensoren, Wiss. Zeitschrift d. Techn. Hochsch. Magdeburg 29 (1985) 7, S. 92–96
/5/ Rebel, B.; Wilhelmi, W.: Trends in image processing architectures and implementations. 1. Internationale Fachtagung „Automatische Bildverarbeitung", Berlin 17./18. Oktober 1985
/6/ Brückner, P.; Geffe, J.: CCD-Linienkamera mit Einchipmikrorechner. radio fernsehen elektronik 35 (1986) 1, S. 14–15
/7/ Frielinghaus, K.-O.; Hecht, S.: Längenmessung mit CCD-Linienkameras. BILD UND TON 37 (1984) 7, S. 205–208
/8/ Dokumentation Durchmessermeßgerät ED 12-12. VEB SKET Magdeburg.
/9/ Heimburger, T. u. a.: Einsatz optoelektronischer Sensoren für meßtechnische Probleme des Schwermaschinenbaus. Wiss. Berichte d. Techn. Hochsch. Leipzig (1986) 2, S. 12–15
/10/ Monecke, J.; Bischoff, W.: Schlaglängenmessung an Verseilgut mit einer CCD-Kamera. Wiss. Beiträge d. IHS Wismar (1984) 4, S. 50–54

✉ KONTAKT ☎

Technische Hochschule Magdeburg,
Sektion Technische Kybernetik und Elektrotechnik,
Boleslaw-Bierut-Platz 5, Magdeburg, 3010;
Tel. 592 131 45

K-1520-kompatible Programmiereinheit

Wolfgang Kabatzke
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik

In dem folgenden Beitrag wird eine K-1520-kompatible Steckeinheit vorgestellt, die das Programmieren von EPROMs mit 1, 2, 4, 8 und 16 KByte ermöglicht. Das angewendete Konzept erlaubt einen universellen Einsatz in allen K-1520-Systemen, so auch im Bürocomputer A 5120 und A 5130. Alle Umschaltfunktionen und Betriebsarteneinstellungen erfolgen softwaremäßig. Mechanische Umschaltelemente wurden nicht eingesetzt. Die Steckeinheit garantiert eine hohe Betriebssicherheit durch entsprechende Schutz- und Überwachungsschaltungen. Ein komfortables Programmpaket für das Betriebssystem SCPX ermöglicht einen effektiven Einsatz.
Unter der Bezeichnung PROMPROG ist die Nachnutzung über das BfN der WPU Rostock möglich.

Einleitung

Trotz großer Fortschritte auf dem Gebiet der Halbleiterspeichertechnik und der Massenspeichertechnik kann auch in modernen Mikrorechnersystemen künftig nicht auf den Einsatz von ROMs und EPROMs verzichtet werden, denn Programmteile zur Initialisierung der Hardware des Systems, zum Laden des jeweiligen Betriebssystems und gegebenenfalls vorhandene einfache Testhilfen (Monitor, Debugger) müssen nach dem

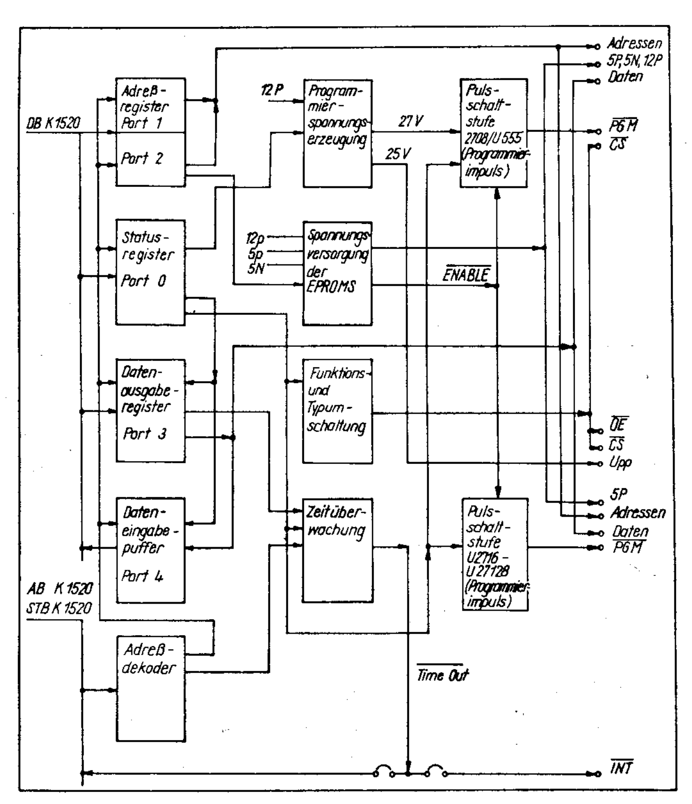
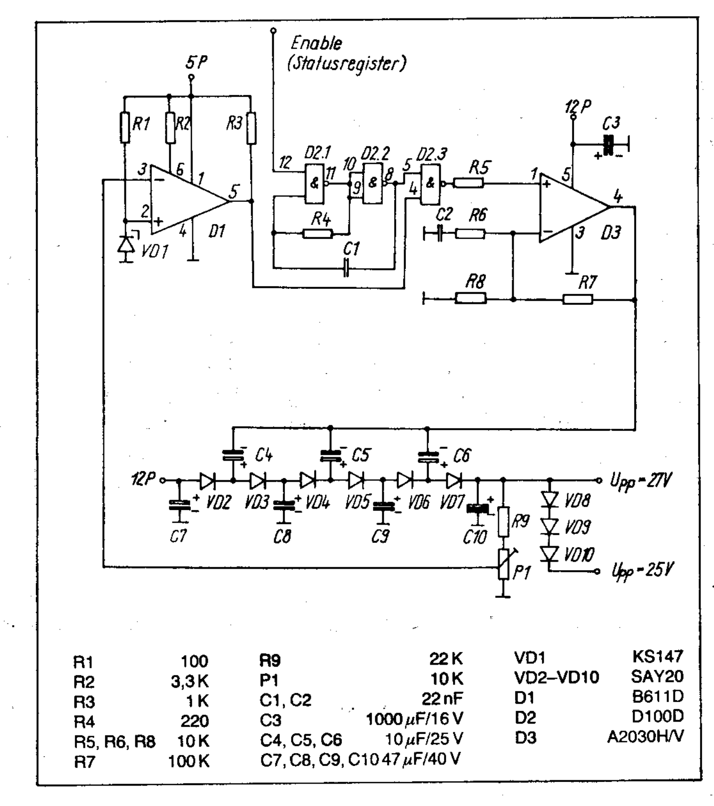


Bild 1 *Prinzipschaltbild der Programmiersteckeinheit*



R1	100	R9	22K	VD1	KS147
R2	3,3K	P1	10K	VD2–VD10	SAY20
R3	1K	C1, C2	22nF	D1	B611D
R4	220	C3	1000µF/16V	D2	D100D
R5, R6, R8	10K	C4, C5, C6	10µF/25V	D3	A2030H/V
R7	100K	C7, C8, C9, C10	47µF/40V		

Bild 2 *Programmierspannungserzeugung*

Einschalten eines Mikrorechnersystems sofort aktivierbar sein.
Vor allem in Entwicklungs- und Testphasen bietet sich vorteilhaft der Einsatz von EPROMs an.
Bei der Konstruktion moderner Mikrorechnerysteme ist auch der Stromversorgung besondere Beachtung zu schenken. Es wird in diesen Systemen auf den Einsatz von nur noch einer Versorgungsspannung (+ 5 V) orientiert. Die Hersteller von Halbleiterspeichern unterstützen diesen Trend durch die Herstellung entsprechender Bauelemente (RAMs, EPROMs).
Für eigene applikative Schritte bei der Entwicklung von Mikrorechnersystemen auf der Basis der Prozessoren U8001/8002 und K 1810 WM86 (I8086) mußte eine Möglichkeit zur Programmierung von 5V-EPROMs geschaffen werden. Für die Auswahl bereits verfügbarer Steckeinheiten wurden folgende Auswahlkriterien herangezogen:

1. Programmierung der EPROMs U 2716 und U 2732
2. perspektivischer Einsatz für die Weiterentwicklung U 2764 und U 27128
3. Programmierung der bisher verwendeten U 555 (2708)
4. Einsetzbarkeit der Steckeinheit in Bürocomputern; das heißt, keine Verwendung von mechanischen Umschaltelementen zur Funktions- und Typenumschaltung
5. Programmierspannungserzeugung aus den vorhandenen Spannungen des Mikrorechnersystems
6. Hohe Betriebssicherheit und leichte Handhabbarkeit, das heißt, die Nutzung muß auch durch Studenten möglich sein.
7. einfache softwaremäßige Anpaßbarkeit an beliebige Betriebssysteme
8. Programmiermöglichkeit von EPROMs für 16-Bit-Mikrorechner.

Eine große Bedeutung kam dabei den Punkten 4. und 6. zu, denn diese widerspiegeln den Haupteinsatzort (Bürocomputer) und den Haupteinsatzzweck (studentische Ausbildung und Forschung). Diese Punkte werden durch keine derzeitig verfügbare Steckeinheit erfüllt, da für alle in Frage kommenden Programmiersteckeinheiten die Umschaltfunktionen per Wahlschalter realisiert sind und ein Einsatz im Bürocomputer dadurch als problematisch anzusehen ist (das Gehäuse des Bürocomputers müßte ständig geöffnet sein, da die Bedienelemente dieser Steckeinheiten hervorstehen; schlechte Zugänglichkeit der Bedienelemente an der Rückseite des Bürocomputers).

Funktionsgruppen der Steckeinheit

Die Steckeinheit läßt sich in folgende Funktionsgruppen untergliedern (siehe Bild 1):

- Adreßregister (PORT1, PORT2)
- Statusregister (PORT0)
- Datenausgaberegister (PORT3)
- Dateneingabepuffer (PORT4)
- Programmierspannungserzeugung
- Spannungsversorgung EPROM
- Funktions- und Typumschaltung
- Zeitüberwachung
- Pulsschaltstufen
- Adreßdekoder und K-1520-Busanpassung.

Bei der Konzipierung dieser Funktionsgruppen wurde auf den Einsatz moderner und platzsparender Bauelemente orientiert und bei der Realisierung der Ports auf den Einsatz von PIOs (U 855) verzichtet. Es gelangten Treiberschaltkreise DS 8282 und DS 8286 zum Einsatz. Durch die günstige Topologie dieser Bauelemente konnte die Topologie der gesamten Steckeinheit positiv beeinflußt werden.
Die Steckeinheit belegt im Adressierungsraum des Mikrorechners 16 E/A-Portadressen. Die hohe Anzahl kommt dadurch zustande, daß außer den 5 Portadressen noch dekodierte Adressen zur Auslösung von Schaltvorgängen (z. B. Register laden) verwendet werden.
Das **Adreßregister,** bestehend aus zwei 8-Bit-Ports DS 8282, übernimmt die Abspeicherung der EPROM-Adressen A0 bis A12. Die drei freien Bits werden zum Schalten der EPROM-Versorgungsspannungen verwendet.
Das **Statusregister** (8-Bit-Port DS 8282) übernimmt die Abspeicherung des Adreßbits A13 und der Steuerbits für die Funktions- und Typumschaltung. Die in das Statusregister einzuschreibende Bitkombination legt fest, ob

- ein EPROM gelesen wird
- ein EPROM programmiert wird oder
- typspezifische Signale (z. B. /OE/ Vpp beim U 2716) aktiviert werden.

Das **Datenausgaberegister** muß die in die EPROMs einzuschreibenden Daten

gepuffert bereitstellen. Hier gelangt ein 8-Bit-Port DS 8282 zum Einsatz. Die Dateneingabe kann ungepuffert erfolgen, hierbei wurde ein 8-Bit-Datenpuffer DS 8286 eingesetzt.
Das Statusregister bildet mit der Funktions- und Typumschaltung das Kernstück dieser Steckeinheit.
In der Funktionsgruppe **Funktions- und Typumschaltung** erfolgen weiterhin
- die Umschaltung der Programmierspannung von + 25 V (U 2716, U 2732 auf + 21 V (U 2764, U 27128)
- die Aktivierung der Signale /OE und /CS und die Festlegung der Datenrichtung (Eingabe/Ausgabe)
- und die Umschaltung typspezifischer Signale.

Bei der Konzipierung der Funktionsgruppe **Programmierspannungserzeugung** wurden folgende Forderungen gestellt:

- hohe Langzeitkonstanz der Programmierspannung (± 0,1 V)
- einfacher und platzsparender Aufbau
- Strombelastbarkeit mindestens 100 mA
- Einsatz moderner Bauelemente
- Realisierung als DC-DC-Wandler.

Der realisierte DC-DC-Wandler ist in Bild 2 dargestellt. Er besteht aus

- einem empfindlichen Komparator
- einem RC-Generator mit Freigabeeingang
- einem Leistungsverstärker und
- einer Spannungsvervielfacherschaltung.

Der RC-Generator schwingt auf einer Frequenz von ca. 15 kHz. Er erzeugt die Ansteuerimpulse für den Leistungsverstärker des DC-DC-Wandlers. Durch die relativ hohe Frequenz des RC-Generators können die Kapazitäten in der Spannungsvervielfacherschaltung gering gehalten werden.
Der Komparator D1 vergleicht über einen Spannungsteiler (R9 und P1) die im DC-DC-Wandler erzeugte Programmierspannung mit der Sollspannung, welche über VD1 erzeugt wird. Dieser Komparator (B 611 D) wird mit TTL-Versorgungsspannung betrieben, da er über seinen Ausgang direkt den nachfolgenden RC-Generator steuert. Als Sollspannungsgeber wurde eine Z-Diode vom Typ KS 147 eingesetzt. Für den Fall, daß die Ausgangsspannung des DC-DC-Wandlers zu klein ist, schaltet der Komparatorausgang auf H-Potential. Dadurch wird der Ausgang des RC-Generators freigegeben, und dessen Impulse gelangen über R6 an den Eingang des Leistungsverstärkers D3. Dieser erzeugt Ausgangsimpulse gleicher Frequenz mit einem Spannungshub von Uss = 12 V. Die Ausgangsimpulse des Leistungsverstärkers gelangen über C4, C5 und C6 auf die Spannungsvervielfacherschaltung des DC-DC-Wandlers und laden die Kondensatoren C7–C10 auf. Hat die Ausgangsspannung einen solchen Wert erreicht, daß die über R9 und P1 abgegriffene Spannung größer wird als die über VD1 erzeugte Sollspannung, schaltet das Ausgangssignal des Komparators auf L-Potential um und sperrt den RC-Generator. Der Ladevorgang der Kapazitäten der Spannungsvervielfacherschaltung ist unterbrochen. Sinkt infolge von Leckströmen der Elektrolytkondensatoren oder durch Stromentnahme die Ausgangsspannung in der Weise ab, daß die über R9 und P1 abgegriffene Spannung unter dem Sollwert liegt, schaltet der Komparatorausgang wieder um auf H-Potential, wodurch ein erneuter Ablauf der beschriebenen Vorgänge eingeleitet wird. Durch das ausgangsspannungsabhängige Schalten des Komparators wird eine geregelte Ausgangsspannung erzeugt. Im praktischen Aufbau wurde eine Restwelligkeit des DC-DC-Wandlers von Uss = 20 mV bei einer Belastung von 100 mA gemessen. Der Maximalwert der Stromentnahme bei dem realisierten Aufbau liegt bei 150 mA. Oberhalb dieses Wertes erfolgt ein rapides Absinken der Ausgangsspannung, da die Kapazitäten der Spannungsvervielfacherschaltung nicht mehr ausreichend nachgeladen werden. Es läßt sich durch Verstellen von P1 ein Wert der Ausgangsspannung von 22 V ... 34 V einstellen. Sollen mit Hilfe dieser Schaltung andere Ausgangsspannungen realisiert werden, so ist eine Veränderung der Frequenz des RC-Generators notwendig bzw. ist die Anzahl der Stufen der Spannungsvervielfacherschaltung zu verändern. Es hat sich als günstig erwiesen, den Leistungsverstärker D3 durch ein kleines Kühlblech von ca. 8 cm^2 zu kühlen. Die Programmierspannung für den EPROM U 555 (2708) beträgt + 27 V ± 0,1 V und wird direkt am Ausgang der Spannungsvervielfacherschaltung abgegriffen. Die Programmierspannung von + 25 V ± 0,1 V wird aus den + 27 V durch eine Reihenschaltung von drei Siliziumdioden (VD 8 – VD 10) erzeugt.
Zum Schutz der Halbleiterspeicherbauelemente während des Programmiervorganges vor Zerstörung infolge zu langer Programmierimpulszeiten, hervorgerufen durch Programierfehler oder durch Programmabstürze, wurde die Funktionsgruppe **Zeitüberwachung** als ein wichtiger Bestandteil mit konzipiert und aufgebaut. Sie besteht aus einem vom Mikrorechner ladbaren 8-Bit-Rückwärtszähler, einem durch den Programmierimpuls freigebbaren RC-Generator und einem Monoflop zur Erzeugung eines Abschalt- und Programmunterbrechungssignals. Der Rückwärtszähler wird vor dem Auslösen jedes Programmierimpulses mit einer Zeitkonstanten geladen, die entsprechend der gewählten Taktfrequenz des RC-Generators von 3,75 kHz die maximale Zeitdauer des Programmierimpulses darstellt. Beim normalen Programmierablauf wird der Rückwärtszähler rechtzeitig, das heißt vor Erreichen des Nulldurchganges, durch die Wegnahme des Programmierimpulses gestoppt. Im Störungsfall läuft der Zähler weiter und gelangt in den Nulldurchgang. Nach Auslösung des Nulldurchgangsimpulses (Übertrag Rückwärts) laufen folgende Vorgänge auf der Programmiersteckeinheit ab:
- Abschalten der Programmierspannungserzeugung
- Abschalten des Programmierimpulses
- Abschalten der EPROM-Versorgungsspannungen
- Desaktivieren aller Ports
- Auslösung eines Programmunterbrechungssignals (NMI oder Interrupt über einen Peripheriebaustein).

Die verwendeten Pulsschaltstufen sind nach den Angaben der EPROM-Hersteller /1/, /2/ konzipiert und aufgebaut (2708 (U 555) nach /3/).

Firmware

Für die Programmiersteckeinheit wurde ein Softwarepaket erstellt, welches sich aus zwei Hauptbestandteilen zusammensetzt:

- betriebssystemunabhängiger Anteil (Gerätetreiber)
- betriebssystemabhängiger Anteil (Bedienerkommunikation, Bedienung Massenspeicher).

Für das Betriebssystem SCPX 1526 ist auf diese Art und Weise ein Programmpaket mit einem Umfang von etwa 8 KByte entstanden. Es enthält folgende Anteile:

- Typauswahl (Auswahl eines gewünschten EPROM-Typs)
- Modusauswahl:

0 EPROM einlesen und Abspeichern auf Massenspeicher (Diskette oder Kassette) und CRC-Berechnung für jeden EPROM
1 EPROM programmieren vom Massenspeicher ohne Leertest mit CRC-Berechnung für jeden EPROM
2 EPROM programmieren vom Massenspeicher mit Leertest und CRC-Berechnung für jeden EPROM
3 Leertest
4 EPROM kopieren mit CRC-Berechnung für jeden programierten EPROM
5 EPROM einlesen in den Programmierpuffer, einzelne Speicherstellen

modifizieren, neuen EPROM mit dem geänderten Bitmuster beschreiben und CRC-Berechnung
6 EPROM-Inhalt auf dem Display ausgeben
7 Einzelbyteprogrammierung für die Typen U 2716–U 27128
8 8086-bezogener Programmierteil (der verwendete Crossassembler erzeugt einen zusammenhängenden Maschinencode, der entsprechend der Speicheraufteilung in High-Bank und Low-Bank aufgesplittert werden muß; der verwendete Crossassembler (IH Wismar Sektion TdE/E) für den U 8000/8001 übernimmt die Aufsplitterung automatisch in seinem Linklauf)

Es wurde eine umfassende Bedienerführung implementiert, die ein einfaches Arbeiten mit diesem Softwarepaket ermöglicht.

Literatur

/1/ INTEL: Memory-design Handbook 1979
/2/ HITACHI: Semiconductor-Data-Book 1981, 1984
/3/ Nicklisch, C.; Winter, W.: Programmieren von EPROMs. rfe 30 (1981) 3, S. 177–179
/4/ von Bechen, P.: Universelle Anschlußbelegung erleichtert Speicheranwendung. Elektronik (1982) 10, S. 75–78

✉ KONTAKT ☎

Wilhelm-Pieck-Universität Rostock,
BfN, Schwaansche Straße 2,
Rostock, 2500; Tel. 36923393

mikroelektronik

VEB Kombinat Mikroelektronik:

Leistung für den Fortschritt

Mit über 59 000 Werktätigen in 22 Betrieben und Einrichtungen gehört der VEB Kombinat Mikroelektronik zu den größten Industriekombinaten der DDR. Entsprechend der Dominanz der Schlüsseltechnologie Mikroelektronik in der ökonomischen Strategie der DDR, mit dem Blick auf das Jahr 2000, kommt dem Kombinat in der Industrie eine entscheidende Rolle bei der gesamtgesellschaftlichen Entwicklung der Deutschen Demokratischen Republik zu. Denn der VEB Kombinat Mikroelektronik ist vor allem verantwortlich für die bedarfsdeckende Bereitstellung von mehr als 1400 Grundtypen aktiver elektronischer Bauelemente für die Anwender-Industrie in allen Volkswirtschaftszweigen. Diese bilden eine wesentliche Grundlage für das Erreichen effektiver Produktions- und Exportsortimente, die Erhöhung der Material- und Energieökonomie, die beschleunigte Automatisierung der Arbeitsprozese sowie die Verbesserung der Arbeits- und Lebensbedingungen der Menschen.
Zum umfangreichen und vielgestaltigen Produktionssortiment gehören Mikroprozessorsysteme mit Verarbeitungsbreiten bis zu 16 Bit, Einchip-Mikrorechner, hochintegrierte Speicherschaltkreise, Analog-Schaltkreise für die industrielle und Konsumgüterelektronik, diskrete Bauelemente wie Dioden, Transistoren, Gleichrichter, mikro-optoelektronische Bauelemente und CCD-Sensorschaltkreise, Bauelemente für die Lichtleiternachrichtentechnik sowie modernste Inline-Farbbildröhren in verschiedenen Dimensionen, Röntgen- und Spezialröhren.
Das Fertigungsprogramm des Kombinates umfaßt aber auch Finalerzeugnisse und Konsumgüter wie Uhren, Taschenrechner, Taschenradios und Radio-Wecker-Kombinationen, Schachcomputer, Kleincomputer und Kopiergeräte.
Dem wachsenden Bedarf der Volkswirtschaft Rechnung tragend, realisieren die Forscher, Technologen, Konstrukteure, Fertigungsingenieure und Facharbeiter des Kombinates beachtliche Zuwachsraten in Entwicklung, Produktion und Absatz.
Eine kleine Auswahl aus dem breiten Angebot an elektronischen Bauelementen wird auf dieser Seite vorgestellt. Ausführlichere Informationen – insbesondere zu Bauelementen der Computertechnik – finden Sie in den folgenden Ausgaben der Mikroprozessortechnik, zum Beispiel in Heft 4/1987 anläßlich des 12. Mikroelektronikbauelementesymposiums.
Sein Leistungsvermögen demonstriert der VEB Kombinat Mikroelektronik auch zur Leipziger Frühjahrsmesse – 15. bis 20. März 1987 – in Halle 15.

Schaltkreise für die Rechen- und Automatisierungstechnik

- Mikroprozessoren U 880/U 8001
Verarbeitungsbreite 8 bzw. 16 Bit, Taktfrequenz 4 MHz
- Grafik-Display Controller U 82720
Adressierbereich 4 Megabit; 1024 × 1024 Bildpunkte in 4 Ebenen; interner Grafik-Symbolgenerator
- 64-KBit-DRAM-Speicherschaltkreis U 2164
Zugriffszeit kleiner 200 ns; eine Versorgungsspannung (5 V)
- 32-KBit-EPROM-Speicherschaltkreis U 2732
Organisation 4 K × 8; Zugriffszeit 350 ns
- Analog-/Digital-Wandler-Schaltkreis C 570/571
Auflösung 8 bzw. 10 Bit; Linearitätsfehler + 1/2 LSB

Bauelemente der Optoelektronik

- Laserdiode für Lichtleiternachrichtenübertragung 850 nm VQ 150
- CCD-Zeile L 133
selbstabtastende, lichtempfindliche Zeile mit 1024 Sensorelementen
- Lichtschachtanzeige VQB 200
16 Segmente, grün
- PIN – Fotodiode SP 106

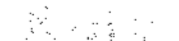
Vorteile:
– störunempfindliche breitbandige Nachrichtenübertragung
– hohe Bildpunktausgabefrequenz
– kompakte optische Baugruppen

Einsatzgebiete:
– Lichtleiternachrichtenübertragung
– Bildaufnahmekameras
– alphanumerische Displays
– Infrarotfernbedienung

Aufsetzbare Bauelemente (SMD)

- Transistoren und Dioden
- unipolare Schaltkreise
- bipolare digitale Schaltkreise
- bipolare analoge Schaltkreise
- sonstige Bauelemente

in SMD-Gehäuseformen SOD, SOT, PCC, SO, CC, QFP.
Bei konsequenter Systemlösung von Bauelement, Bestückungstechnik und Fertigungstechnologie erreichbare Vorteile in der Geräteindustrie:
– 3- bis 10fach geringere Produktionskosten
– höhere Qualität und Funktionssicherheit
– Miniaturisierung von Fläche und Volumen
– höhere mechanische Belastbarkeit durch die Leiterplatte
– technische Vorteile in der Hochfrequenz-Anwendung.



Programmierung in C

(Teil II)

Dr. Thomas Horn
Ingenieurhochschule Dresden
Sektion Informationsverarbeitung

Nachdem im Teil I dieser Artikelserie (s. Mikroprozessortechnik 1/87) eine Einführung in die Lexik der Programmiersprache C gegeben und die grundsätzlichen einfachen Datentypen und Speicherklassen behandelt wurden, werden nun die Operatoren und Ausdrücke sowie die Steuerstrukturen erläutert. Sowohl die Operatoren und Ausdrücke als auch die Steuerstrukturen widerspiegeln maßgeblich das moderne und leistungsfähige Sprachkonzept der Programmiersprache C, das zu ihrer weiten Verbreitung in der Systemprogrammierung beigetragen hat.

3. Operatoren, Ausdrücke und einfache Anweisungen

In C werden drei Klassen von Operatoren unterschieden:

- einstellige Operatoren,
- zweistellige Operatoren und
- Zuweisungsoperatoren.

Ausdrücke (*expressions*) werden durch Anwendung der Operatoren auf Variablen und Konstanten gebildet. Da in Ausdrücken oftmals mehrere Operatoren Anwendung finden, ist ihre Priorität und bei gleicher Priorität die Abarbeitungsreihenfolge sehr wichtig. Einstellige Operatoren haben die höchste und Zuweisungsoperatoren die niedrigste Priorität. Zur Änderung der Reihenfolge der Abarbeitung der Operatoren ist eine Klammerung mit runden Klammern zulässig.
Eine höhere Priorität als die einstelligen Operatoren haben der Funktionsaufruf *identifikator***( )**, die Spezifikation von Elementen eines Feldes *identifikator***[ ]** und die Auswahl von Elementen einer Struktur (— > bzw. .)
Weiterhin gibt es noch einen Kommaoperator zur Aufzählung von Ausdrükken (Aufzählungsoperator). Er liegt in der Priorität noch hinter den Zuweisungsoperatoren. Die Gesamteinordnung aller Operatoren in ein Prioritätssystem unter Angabe der Abarbeitungsreihenfolge ist in Bild 1 enthalten.

3.1. Die einstelligen Operatoren

Einstellige Operatoren stehen in der Regel vor dem Operanden und werden von rechts nach links abgearbeitet, wenn vor einem Operanden mehrere Operatoren angegeben werden. Folgende einstellige Operatoren sind definiert:

- ***** Indirektadressierung, z. B. ***pa** bedeutet, daß der Inhalt von **pa** als Operandenadresse zur Adressierung einer Variablen benutzt wird.
- **&** Adreßoperator, z. B. **&A [1] [2]** bedeutet Adresse des Feldelements **A [1] [2]**.
- **—** Zweierkomplement, z. B. **—alpha**
- **~** Einerkomplement, z. B. **~alpha**
- **!** logische Negation (Logische Variablen sind vom Typ **int** oder **char**. Der Wert 0 bedeutet *false*, und ein Wert ungleich 0 bedeutet *true*).
- **++** Inkrementoperator
- **——** Dekrementoperator.

Der Inkrement- und der Dekrementoperator können vor und nach dem Operanden stehen. Vor dem Operanden geben sie ein Inkrementieren bzw. Dekrementieren vor der Speicheroperation an und nach dem Operanden das Entsprechende nach der Speicheroperation. Die Stellung der Inkrement- und Dekrementoperatoren hat damit nur bei den Feld- und Zeigeroperatoren Bedeutung; z. B. wird bei **text[i++]** auf das Zeichen **text[i]** zugegriffen, und anschließend wird der Index **i** inkrementiert, während bei **text[++i]** zugegriffen wird. Variablen vom Typ **int** und **char** werden um 1 inkrementiert bzw. dekrementiert. Zeigervariablen werden entsprechend der physischen Speicherlänge des Objekts inkrementiert bzw. dekrementiert.
Zur Realisierung einer maschinennahen, aber doch portablen Programmierung kann der einstellige Operator

sizeof (*type***)** oder **sizeof** *identifikator*

verwendet werden, der die physische Speicherlänge von Datentypen oder Variablen (Felder, Strukturen usw.) ermittelt, z. B.

sizeof(int) oder **sizeof FELD**.

Ein weiterer einstelliger Operator, die explizite Typangabe **(***type***)**, kann benutzt werden, um eine bestimmte Typumwandlung zu erzwingen.

Beispiel:
sqrt((double)n)
a = (int)sqrt(7.0)

Die Funktion **sqrt** erwartet einen Operanden vom Typ **double**. Wenn **n** in einem anderen Typ vorliegt, wird durch die explizite Typangabe eine Umwandlung in das **double**-Format erzwungen. Im zweiten Beispiel wird das Ergebnis der Funktion **sqrt** in eine Integerzahl umgewandelt.

3.2. Die zweistelligen Operatoren

Zweistellige Operatoren werden nachfolgend in Gruppen nach fallender Priorität erläutert. Innerhalb der Gruppen haben die Operatoren eine gleiche Priorität und werden von links nach rechts abge-

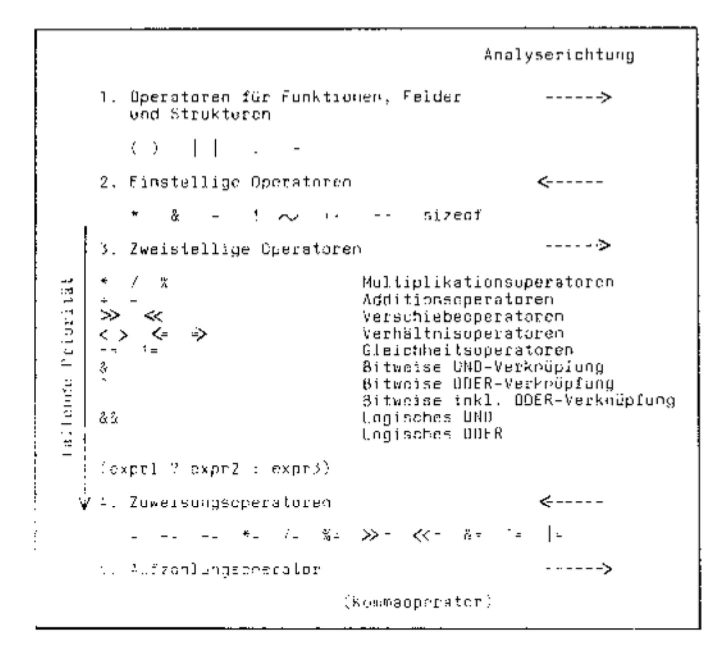


Bild 1 Prioritäten und Analyserichtung der Operatoren

arbeitet, wenn in einem Ausdruck mehrere zweistellige Operatoren angegeben sind.

- **Multiplikationsoperatoren:**
  - `*` Multiplikation
  - `/` Division
  - `%` Rest von der ganzzahligen Division
- **Additionsoperatoren:**
  - `+` Addition
  - `−` Subtraktion
- **Verschiebeoperatoren:**
  - `>>` Rechtsverschiebung, z. B. **alpha>>2** (2 Bit nach rechts)
  - `<<` Linksverschiebung, z. B. **alpha<<A0** (A0 Bit nach links)
- **Verhältnisoperatoren:**
  - `<` kleiner als
  - `>` größer als
  - `<=` kleiner oder gleich
  - `>=` größer oder gleich
- **Gleichheitsoperatoren:**
  - `==` gleich
  - `!=` ungleich
- **Bitweise UND-Verknüpfung:**
  - `&` Die einzelnen Bits der Variablen vom Integertyp werden bitweise konjunktiv (logisches UND) verknüpft.
- **Bitweise exklusive ODER-Verknüpfung:**
  - `^` Die einzelnen Bits der Variablen vom Integertyp werden bitweise entsprechend der Funktion Antivalenz (exklusives logisches ODER) verknüpft.
- **Bitweise inklusive ODER-Verknüpfung:**
  - `|` Die einzelnen Bits der Variablen vom Integertyp werden bitweise disjunktiv (inklusives ODER) verknüpft.
- **Logisches UND:**
  - `&&` Die Werte der Variablen werden als logische Werte *true* oder *false* interpretiert und konjunktiv (logisches UND) verknüpft.
- **Logisches ODER:**
  - `||` Die Werte der Variablen werden als logische Werte *true* oder *false* interpretiert und disjunktiv (logisches ODER) verknüpft.
- **Bedingter Operator:**
  expression1 **?** *expression2* **:** *expression3*

Wenn der Ausdruck *expression1 true* (ungleich Null) ist, so gilt *expression2*, andernfalls *expression3*.

Beispiel:
V= a0 < a1 ? a0 : a1;

Der Variablen **V** wird entweder **a0** oder **a1** zugewiesen, in Abhängigkeit davon, welche Variable den kleineren Wert hat.

3.3. Die Zuweisungsoperatoren

Der generelle Zuweisungsoperator ist wie in anderen Programmiersprachen das Gleichheitszeichen (=). Da sehr häufig zweistellige Operationen mit einer Zielvariablen vorkommen, werden zur Erhöhung der Prägnanz weitere Zuweisungsoperatoren zugelassen:

```
+=  -=  *=  /=  %=  <<=
>>=  &=  ^=  |=
```

Auf der linken Seite der Zuweisungsoperatoren muß immer ein Ausdruck stehen, dem ein Wert zugewiesen werden kann, also eine einfache Variable oder ein Adreßausdruck mit dem Indirektoperator (vergleiche auch 1.5.).

Beispiele:
a+=2; /* entspricht a=a+2 */
b<<=5; /* entspricht b=b>>5 */

Außerdem werden in einer Anweisung mehrere Zuweisungsoperatoren zugelassen, die dann von rechts nach links abgearbeitet werden, zum Beispiel

a+=b*=c=d>>=2;

bedeutet:

a=a+b*(d>>2);
b=b*(d>>2);
c=d>>2;
d=d>>2;

3.4. Anweisungen

Ausdrücke werden zur Anweisung (*statement*), wenn sie durch ein Semikolon abgeschlossen werden, zum Beispiel

a=b=0;
j++;
putc(n);
; /* Leeranweisung */

In C wird eine Anweisung durch ein Semikolon beendet, im Gegensatz zu PASCAL, wo ein Semikolon zwei Anweisungen trennt. Durch die Verwendung von geschweiften Klammern können mehrere Anweisungen zu einer Verbundanweisung (*compound statement*) zusammengefaßt werden, die – syntaktisch gesehen – wie eine Anweisung behandelt wird. Nach der rechten geschweiften Klammer steht niemals ein Semikolon. Der Unterschied zwischen einer Verbundanweisung und einem Block besteht darin, daß in einer Verbundanweisung keine neuen Variablen deklariert werden.

4. Steuerstrukturen

Steuerstrukturen werden innerhalb von Funktionen verwendet, um die Reihenfolge der Abarbeitung von Anweisungen festzulegen. In den einzelnen Steuerstrukturen darf in der Regel eine Anweisung (*statement*) stehen. Sollen mehrere Anweisungen abgearbeitet werden, so sind diese durch geschweifte Klammern zu einer Verbundanweisung (*compound statement*) zusammenzufassen. Nach der linken geschweiften Klammer dürfen Deklarationen für neue Variablen stehen, so daß anstelle einer einzelnen Anweisung tatsächlich auch immer ein Block stehen darf.[3]
Unter Verwendung der weitverbreiteten Struktogrammtechnik zur grafischen Darstellung der Programmstrukturen kann eine Funktion im allgemeinen als eine Sequenz von *Strukturblöcken* (Bild 2) dargestellt werden. Im Sinne von C ist jeder *Strukturblock* eine einzelne Anweisung, eine Verbundanweisung oder ein Block.

[3] siehe Fußnote 2 in Teil I

Strukturblock 1
Strukturblock 2
...
Strukturblock n

Bild 2
Sequenz von Strukturblöcken

if(cond) true	false
Strukturblock	Strukturblock

Bild 3
Vollständige Alternative

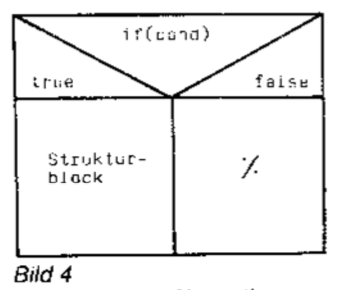


Bild 4
Unvollständige Alternative

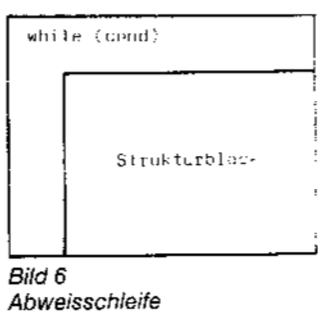


Bild 6
Abweisschleife

4.1. Die bedingte Anweisung

Die bedingte Anweisung (Alternative) ist als **if-else**-Anweisung realisiert, wobei der **else**-Teil wahlfrei ist. Bild 3 zeigt das Struktogramm für die vollständige und Bild 4 für die unvollständige Alternative. Die Syntax der bedingten Anweisung ist:

if (*expression***)** *statement1* **else** *statement2*

Der Ausdruck *expression* wird berechnet und als Verzweigungsbedingung *cond* bewertet. Ist das Ergebnis ungleich 0 (*true*), so wird *statement1* (linker Strukturblock) ausgeführt. Wenn das Ergebnis gleich 0 (*false*) ist, so wird, falls vorhanden, die Anweisung *statement2* im **else**-Teil (rechter Strukturblock) ausgeführt.

Beispiel:

```
if (n-1) f=f*(n-1); else f=1;
oder
if (n==1) f=1; else f=f* (n-1);
```

Sowohl im **if**-Teil als auch im **else**-Teil dürfen im auszuführenden Strukturblock weitere **if**-Anweisungen stehen. Da der **else**-Teil wahlfrei ist, besteht bei geschachtelten **if**-Anweisungen die Gefahr, daß die **else**-Teile in der Programmnotation nicht eindeutig zugeordnet werden können. Vom Compiler wird deshalb ein **else**-Teil immer der letzten **if**-Anweisung zugeordnet, die noch keinen **else**-Teil hat.

Beispiel:

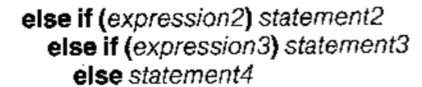
```
if (n<0) if (x<y) z=x;
   else z=y;
```

Der **else**-Teil entspricht der zweiten **if**-Anweisung. Eine Zuordnung zur ersten **if**-Anweisung kann durch geschweifte Klammern erzwungen werden:

```
if (n<100) {if (x<y) z=x;}
else z=y;
```

Wesentlich günstiger ist die Anordnung weiterer **if**-Anweisungen im **else**-Teil, z. B.:

if (*expression1***)** *statement1*



else if (*expression2***)** *statement2*
  else if (*expression3***)** *statement3*
    else *statement4*

Diese Folge von **if**-Anweisungen ist die allgemeingültige Möglichkeit, unter vielen Alternativen eine Möglichkeit herauszusuchen. In der angegebenen Reihenfolge werden die Ausdrücke berechnet, und falls ein Ausdruck *true* ist, wird die entsprechende Anweisung ausgeführt. Der letzte **else**-Teil behandelt den Fall, daß keine der vorherigen Bedingungen zutraf. Er kann dazu benutzt werden, den ‚allgemeinen' Fall oder eine ‚unmögliche' Bedingung zu realisieren. Falls nicht erforderlich, kann der letzte **else**-Teil auch entfallen.

4.2. Die Fallauswahl

Die Fallauswahl (Bild 5) ist eine Erweiterung der Alternative auf mehr als zwei Fälle. Sie wird über die **switch**- und **case**-Anweisungen realisiert. In Abhängigkeit vom Wert der Bedingung *cond* (Schalterausdruck *expression*) kann einer von mehreren folgenden Strukturblöcken ausgewählt werden. Die **switch**-Anweisung hat folgende Form:

switch (*expression***)** *statement*

Das Resultat des Schalterausdrucks muß ein Integer-Wert sein. Die abhängige Anweisung *statement* ist zweckmäßigerweise eine Verbundanweisung. Jeder Anweisung innerhalb der Verbundanweisung kann eine beliebige Anzahl von **case**-Marken vorausgehen:

case *constant-expression***:**

Nach dem Schlüsselwort **case** muß ein Integer-Wert stehen, der durch einen Konstantenausdruck vorgegeben werden kann; das heißt, der Ausdruck muß zur Compilierzeit berechenbar sein. Jeder Konstantenausdruck muß einen anderen Integer-Wert ergeben.
Für alle übrigen, nicht spezifizierten Fälle wird die **case**-Marke

default:

verwendet.
Wenn für einen Schalterwert keine entsprechende **case**-Marke vorhanden und auch keine **default**-Marke spezifiziert ist, so wird die abhängige Anweisung nicht ausgeführt und die nächste Anweisung nach der **switch**-Anweisung abgearbeitet.
Die **case**-Marken selbst haben keinen Einfluß auf die sequentielle Abarbeitung der Anweisungen in der Verbundanweisung. Sie stellen lediglich verschiedene Einsprungmarken dar. Zum Verlassen eines **case**-Teils wird normalerweise die **break**-Anweisung benutzt. Die **case**-Marke spezifiziert somit den Anfang und die **break**-Anweisung das Ende eines Strukturblockes.

Hinweis:

Da über die **case**-Marken und nicht über die geschweifte Klammer der Eintritt in die Verbundanweisung der **switch**-Anweisung erfolgt, werden keine Laufzeitinitialisierungen von Variablen ausgeführt. Aus diesem Grunde sollten in der Verbundanweisung keine Variablen über die Speicherklasse **auto** oder **register** initialisiert werden.

Beispiel:
Zählen der Leerzeichen, Punkte, Kommas und Zeilenvorschübe in einem Text.

```
main()
{
static int nl=0, np=0, nk=0,
nz=0, nr=0; char c;
while ((c=getchar())!=EOF)
switch(c) {
case ' ': nl++;break;
case '.': np++;break;
case ',': nk++;break;
case '\n': nz++;break;
default:  nr++;break;
}
printf("\nL=%d, P=%d, K=%d,
Z=%d, Rest=%d\n", nl, np,
nk, nz, nr);
}
```

Die Funktion **getchar** stellt immer das nächste Zeichen aus dem Terminalpuffer zur Verfügung. Die Eingabe in den Terminalpuffer erfolgt mittels Drücken der (alphanumerischen) Tasten des Terminals und wird mit der RETURN-Taste abgeschlossen. Das Zeichen **EOF** (End of File) wird durch das Steuerzeichen

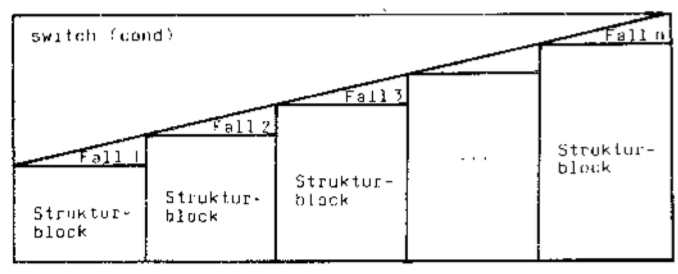


Bild 5 Fallauswahl

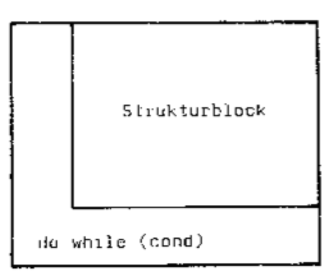


Bild 7 Nichtabweisschleife

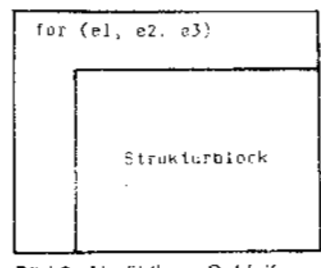


Bild 8 Abzählbare Schleife

CTRL/Z und RETURN eingegeben. Die **while**-Anweisung wiederholt die **switch**-Anweisung bis zum **EOF**.

4.3. Die Abweisschleife

Die Abweisschleife (Bild 6) wird über die **while**-Anweisung realisiert. Sie hat folgende Syntax:

while (*expression***)** *statement*

Der Ausdruck *expression* wird berechnet. Ist sein Wert ungleich 0 (*true*), so wird die abhängige Anweisung *statement* ausgeführt und die **while**-Anweisung wiederholt. Ist der Wert des Ausdrucks 0 (*false*), so wird die Schleife beendet. Bei der Wiederholung der **while**-Anweisung wird der Ausdruck *expression* neu berechnet. Ist die Anweisung eine Verbundanweisung, so kann mittels **continue**-Anweisung vorzeitig zur Wiederholung der **while**-Anweisung übergegangen werden. Durch eine **break**-Anweisung kann die **while**-Anweisung vorzeitig verlassen werden.

Beispiel:
Berechnen der Fakultät **f** einer Zahl **i**.

```
f=1; i=6;
while (i) {f=f*i;--i;}
oder
while (1) {if(i<=1)break;
f*=i;i--;}
```

Im zweiten Beispiel würde die **while**-Anweisung durch die Bedingung **(1)** unendlich oft wiederholt werden. Zum Verlassen der Schleife wird die **break**-Anweisung benutzt.

4.4. Die Nichtabweisschleife

Die Nichtabweisschleife (Bild 7) wird als **do-while**-Anweisung realisiert. Sie hat folgende Syntax:

do *statement* **while (***expression***);**

Zuerst wird die Anweisung *statement* ausgeführt und anschließend der Ausdruck *expression* berechnet. Ist der Ausdruck ungleich 0 (*true*), so wird die Schleife wiederholt. Wenn der Ausdruck gleich 0 (*false*) ist, so wird die Schleife beendet. Ist die Anweisung eine Verbundanweisung, so kann mit einer **continue**-Anweisung vorzeitig zur Wiederholung der **do**-Schleife übergegangen werden, während mittels **break**-Anweisung die Schleife vorzeitig verlassen wird.

Beispiel:
Berechnen der Fakultät **f** einer Zahl **i**.

```
f=1; i=6; do f=f*i;
while (--i==0);
```

4.5. Die verallgemeinerte abzählbare Schleife

Die verallgemeinerte abzählbare Schleife (Bild 8) wird durch die **for**-Anweisung realisiert. Sie hat folgende Syntax:

for (*expression1***;***expression2***;***expression3***)** *statement*

Die **for**-Anweisung kann als eine äquivalente **while**-Schleife wie folgt dargestellt werden:

*expression1***;while (***expression2***)**
{*statement expression3***;**}

Der Ausdruck *expression1* dient zur Initialisierung der Schleife. Der Ausdruck *expression2* wird vor jeder Ausführung der abhängigen Anweisung *statement* berechnet und bewirkt bei ungleich 0 (*true*) die Ausführung der abhängigen Anweisung. Bei gleich 0 (*false*) wird die Schleife beendet. Der Ausdruck *expression3* wird typischerweise zur Berechnung des nächsten Wertes der Laufvariablen verwendet.
Jeder Ausdruck kann in der Anweisung ausgelassen werden. Bei ausgelassenem Ausdruck *expression2* gilt **while (1)**. Die Schleife muß dann mit einer **break**-, **goto**- oder **return**-Anweisung verlassen werden. Fehlen die Ausdrücke *expression1* und *expression3*, so gelten sie als nicht spezifiziert.

Beispiel:
Suchen des ersten Kommas im Textpuffer **text [100]**:

```
for(i=0;text[i]!='\0';i++)
if(text[i]==',')break;
oder
for(i=-1;text[++i]!='\0';)
if(text[i]==',')break;
oder
for(i=0;text[i]!='\0'
&(text[i]!=',';i++);
```

Der Textpuffer **text[ ]** ist ein Feld. Der Index **i** wird ab Null beginnend gezählt. Mit **text [i]** wird auf das **i**-te Zeichen des Textpuffers zugegriffen. Der Ausdruck **text[++i]** gibt an, daß vor dem Zugriff auf das **i**-te Zeichen der Wert von **i** um Eins erhöht wird. Das Zeichen **\0** gibt das Ende der Zeichenkette an.

4.6. Die Fortsetzungsanweisung

Die Anweisung
continue;
kann sich in den abhängigen Anweisungen einer **while**-, **do**- oder **for**-Anweisung befinden und bewirkt den vorzeitigen Übergang zur nächsten Schleifenausführung, das heißt, es wird mit dem Test auf die Schleifenwiederholung fortgesetzt. Bei der **for**-Anweisung wird aber vorher noch der Ausdruck *expression3* berechnet.

4.7. Die Unterbrechungsanweisung

Die Anweisung

break;

kann sich in den abhängigen Anweisungen einer **while**-, **do**- oder **for**-Anweisung befinden und bewirkt das vorzeitige bedingungslose Verlassen der Schleife, das heißt, es wird mit der nächsten Anweisung, die nach der Schleife folgt, die Programmabarbeitung fortgesetzt.

4.8. Die Sprunganweisung

Die Anweisung

goto *label***;**

bewirkt einen Sprung zu einer Marke *label*. Die Marke muß sich in einem Block der gleichen Funktion befinden. Marken sind *Identifikatoren*, die vor einer beliebigen Anweisung definiert sind:

*label***:**

Vor einer Anweisung kann eine beliebige Anzahl Marken stehen. Der Geltungsbereich der Marken erstreckt sich über die ganze Funktion. Eine Ausnahme sind verschachtelte Blöcke einer Funktion, in denen der gleiche Identifikator neu als Marke definiert wird. Im allgemeinen kann ein Programm ohne die **goto**-Anweisung geschrieben werden. Da die Anwendung der **goto**-Anweisung den Prinzipien der strukturierten Programmierung widerspricht, sollte man auf ihre Anwendung im Interesse einer klaren Programmstruktur verzichten.

Beispiel:
In diesem Beispiel soll gezeigt werden, wie die **goto**-Anweisung sinnvoll in der Systemprogrammierung eingesetzt werden kann, um bei Fehlerzuständen mehrere ineinander geschachtelte Strukturblöcke effektiv zu verlassen.

```
for(...)
{
while(...)
  switch(...) {
  case...:
  case...:...
          if(rc<0) goto ERROR;
          ...
  default:...
  }
}
...
ERROR: /*Fehleranalyse-
routine*/
...
```

Entwicklungsunterstützung für 16-Bit-Mehrmikrorechnersystem

Lutz Dorfmüller,
Dr. Hans-Günter Despang
Zentralinstitut für Kybernetik und Informationsprozesse der AdW der DDR

Zur effektiven Entwicklung von Mikrorechnersystemen sind angepaßte Werkzeuge für jeden Entwicklungsschritt erforderlich. Zum Test von größeren Programmen auf bereits funktionsfähiger Hardware dienen Monitore. In der Anfangsphase der Entwicklung eines Mikrorechnersystems stehen noch keine Peripheriesteuerungen zur Verfügung. Dann werden Monitore benötigt, die eine Kopplung zu einem Cross-Entwicklungssystem besitzen. Mittels einer solchen Kopplung ist die Peripherie des Cross-Systems nutzbar, und die Programmentwicklung kann effektiver gestaltet werden. Einen leistungsfähigen Monitor für I16-Systeme mit einer Kopplung zu einem 8-Bit-Entwicklungssystem und seine Nutzung an einem Zweirechnersystem beschreibt der folgende Beitrag.

Einführung

Bei der Entwicklung von Mikrorechnerlösungen für dedizierte Anwendungsfälle (z. B. Steuerung eines Meßgerätes), bei der Entwicklung von modularen Hard- und Software-Baugruppensystemen und auch bei der Entwicklung von kompletten Mikrorechnern wird eine gut ausgebaute und in ihrer Wirkungsweise aufeinander abgestimmte Entwicklungsunterstützung benötigt. Derzeit wird bereits eine breite Palette von Systemen für diesen Zweck auf dem internationalen Markt angeboten (vgl. Literatur). Dabei ist im wesentlichen eine Einordnung in folgende Klassen möglich:
- Logikanalysatoren
- In-Circuit-Emulatoren
- Monitore
- Multitask-orientierte Debugger
- Entwicklungssysteme.

Logikanalysatoren

Logiganalysatoren unterstützen die Hardware-Inbetriebnahme von Mikrorechnersystemen auf dem Logikniveau. Signale können an beliebigen Stellen auf Leiterkarten und Bussen abgegriffen und als Impulsfolgediagramm oder in zusammengefaßter Form (hexadezimal) angezeigt werden. Auf dieser Ebene geht es noch nicht um die Signalzusammenfassung zu mikrorechnertypischen Darstellungen wie z. B. zur mnemonischen Kodierung der über den Bus übertragenen Befehle. Viele Logikanalysatoren weisen jedoch Zusätze auf (Rückübersetzer), die eine solche Darstellung erlauben. Diese Zusätze erweitern das Einsatzfeld der Logikanalysatoren wesentlich, sind aber immer auf einen bestimmten Prozessortyp zugeschnitten. Das Einsatzgebiet der Logikanalysatoren ist aber schwerpunktmäßig die Inbetriebnahme der Logik auf einzelnen Leiterkarten.

In-Circuit-Emulatoren

Zur Inbetriebnahme eines Gesamtsystems (z. B. dediziertes System oder kompletter Rechner) werden In-Circuit-Emulatoren (ICEs) verwendet. Sie übernehmen die Überwachung und die Aufzeichnung des gesamten Verkehrs des Prozessorschaltkreises mit seiner Umwelt. In die Steckfassung des Prozessorschaltkreises des zu entwickelnden Zielsystems wird dazu das Emulatorkabel gesteckt, das die Verbindung zum ICE herstellt, der auch die Funktion des Prozessors mit übernimmt. Die Anwendung eines ICEs setzt eine logisch funktionsfähige Prozessorumgebung im Zielsystem voraus und erlaubt, ausgehend von diesem funktionsfähigen Hardwarekern, den weiteren Ausbau und die Inbetriebnahme des Zielsystems. Zur Inbetriebnahme der peripheren Einheiten ist nach wie vor der Einsatz des Logikanalysators erforderlich, um Fehler in der Logik erkennen und beseitigen zu können.
Neben der Hardwareinbetriebnahme dienen Emulatoren insbesondere auch der Inbetriebnahme von einfachen, hardwarenahen Programmen. Als Besonderheit tritt auf, daß Software schon getestet werden kann, wenn das Zielsystem hardwaremäßig noch nicht komplett aufgebaut ist. Fehlende Ressourcen (z. B. Speicher) werden im zugehörigen Entwicklungssystem „geborgt". Nachteilig wirkt sich aus, daß sich bei der Anwendung von ICEs oft Einschränkungen in der Beobachtbarkeit der Echtzeiteigenschaften der Software ergeben. Programme im geborgten Speicher laufen z. B. langsamer ab als im echten Speicher des Zielsystems.
Ist die Funktion der Hardware im Zielsystem gesichert und sind die hardwarenahen Programme getestet, macht sich eine solch detaillierte Beobachtung der Prozessorsignale, wie sie der ICE ermöglicht, nicht mehr erforderlich. Es geht nun beim weiteren Systemausbau um die Entwicklung und Inbetriebnahme größerer Softwarepakete auf einer funktionsfähigen Hardware. Dazu werden Monitore eingesetzt.

Monitore

Die Softwareentwicklung kann in Abhängigkeit vom erreichten Hardwareausbau und späteren Verwendungszweck des Zielsystems entweder direkt auf diesem oder auf einem separaten Mikrorechner-Entwicklungssystem stattfinden. Ein separates Entwicklungssystem wird dann eingesetzt, wenn das Zielsystem noch nicht alle für die Programmentwicklung erforderlichen Ressourcen enthält (z. B. externe Speicher) oder wegen einer dedizierten Anwendung nicht erhalten soll. In diesem Fall macht sich das Laden der auf dem Entwicklungssystem erstellten Programme in das Zielsystem erforderlich, wozu der Monitor zusätzliche Kommandos enthält.
Monitore haben den Nachteil, daß sie gewöhnlich nur den linearen Programmablauf, wie er auf dem Prozessor stattfindet, überwachen können. Größere Programmsysteme zeichnen sich aber meist durch eine Zerlegung in mehrere Tasks aus, die unter der Steuerung eines Multitask-Betriebssystems laufen. Zu ihrer Inbetriebnahme werden multitaskorientierte Debugger eingesetzt /1/.

Multitaskorientierte Debugger

Ein multitaskorientierter Debugger ist eine Software zur Inbetriebnahme von Multitasksystemen. Die zur Verfügung gestellten Hilfsmittel sind taskorientiert, so daß das Verhalten einzelner Tasks beobachtet und beeinflußt werden kann, ohne das Verhalten anderer Tasks zu stören. Der Überblick über das Verhalten aller Tasks ist ebenso möglich wie die Veränderung des Zustandes einzelner Tasks.

Entwicklungssysteme

Obwohl in der anfänglichen Übersicht die Mikrorechner-Entwicklungssysteme als separater Punkt genannt worden sind, sei hier darauf verwiesen, daß Entwicklungssysteme meist den Hintergrund (Basisrechentechnik) für alle anderen Entwicklungshilfen darstellen. Sie dienen einerseits zur Programmentwicklung, sind aber zusätzlich mit Hard- und Softwarelösungen zur Inbetriebnahme kompletter Systeme ausgestattet (z. B. I^2ICE-Logikanalyse und In-Circuit-Emulation auf INTEL-Entwicklungssystemen /2/). In letzter Zeit werden anstelle der meist firmenspezifischen Entwicklungssysteme verstärkt die weiter verbreiteten Personalcomputer oder auch Kleinrechner eingesetzt /3/, /4/, /5/, /6/.



Dr. Hans-Günter Despang (36) studierte 1970 bis 1974 an der TU Dresden, Sektion Informationstechnik. Er diplomierte 1974 mit einer Arbeit über Fehlererscheinungen an Digitalfiltern. Von 1974 bis 1981 arbeitete er als wissenschaftlicher Mitarbeiter am Zentralinstitut für Kybernetik und Informationsprozesse der AdW der DDR auf dem Gebiet der industriellen binären Steuerungstechnik mit Mikrorechnern. Auf diesem Gebiet promovierte er 1982 mit einer Arbeit über die steuerungstechnische Fachsprache „SGM". Seit 1982 leitet er eine Abteilung, welche auf dem Gebiet der Mehrmikrorechnersysteme arbeitet.

Dipl.-Ing. Lutz Dorfmüller studierte von 1972 bis 1976 an der TU Dresden, Sektion Informationstechnik. Die Diplomarbeit behandelte Entwicklung und Aufbau eines Prüfgerätes für einen LSI-Schaltkreis. Anschließend arbeitete er am ZKI der AdW zunächst auf dem Gebiet der Sensortechnik.
Zur Zeit beschäftigt er sich mit speziellen Hardwareaufgaben für 8-Bit- und 16-Bit-Systeme.



Alle Entwicklungshilfsmittel sollen der Forderung nach Anwenderfreundlichkeit genügen. Ein wichtiger Aspekt dabei ist die Arbeit auf Hochsprach-Niveau wie z. B. beim PSCOPE 86 (in /2/). Darunter ist zu verstehen, daß die Hilfsmittel für die Programminbetriebnahme die Verwendung derselben symbolischen Bezeichner (Marken) gestattet, die der Softwareentwickler bei der Programmentwicklung verwendet hat. Das setzt einen fein aufeinander abgestimmten Softwareentwicklungs- und Inbetriebnahmeprozeß voraus, so daß die symbolischen Bezeichner in den Lademodulen nach wie vor vorhanden sind und von den Entwicklungswerkzeugen weiterverwendet werden können.

Monitor für 16-Bit-Systeme auf der Basis I16

Im folgenden wird ein bei den Autoren für Arbeiten zu einem Mehrmikrorechner-Betriebssystemn /7/ eingesetzter Monitor näher beschrieben. Dieser Monitor zeichnet sich durch eine Kopplung zu einem Entwicklungssystem aus. Er lehnt sich an das iSBC 957B iAPX 86,88 Interface- and -Execution-Package an /8/ und ist ursprünglich nur für Einrechnerarbeit ausgelegt. Mit einfachen Mitteln konnte dieser Monitor auch für die Arbeit an einem Zweirechner-System effektiv eingesetzt werden.
Er ist ein komfortables Werkzeug zum Programmtest für 16-Bit-Systeme auf der Basis der Schaltkreisfamilie I16. Die für die Arbeit mit diesem Monitor erforderliche Minimalkonfiguration besteht aus einer ZVE, 4 K Byte RAM, 16 K Byte PROM und einer seriellen Schnittstelle zum Anschluß eines Terminals. Ein Beispiel einer Minimalkonfiguration mit der ZVE des S-700-Systems /9/ zeigt Bild 1. Mit der Kopplung zwischen 16-Bit-Anwender- und 8-Bit-Entwicklungssystem ergibt sich eine Konfiguration nach Bild 2. In diesem Fall wird der Monitor über die Konsole des Entwicklungssystems bedient. Mit dieser Kopplung erweitert sich der Funktionsumfang des Monitors wesentlich. Die Ressourcen des 8-Bit-Systems (Diskettenstation, Bedienkonsole und zugehörige Systemroutinen) sind nutzbar, auch wenn am 16-Bit-System noch keine Treiber zur Verfügung stehen. Außerdem können Programme und Baugruppen gleichzeitig entwickelt werden, wenn geeignete Cross-Software zur Verfügung steht. Ist das Anwendersystem dann funktionsfähig, können die fertigen Programme über die Kopplung zum Anwendersystem übertragen und dort getestet werden. Die Programmentwicklung wird dadurch wesentlich erleichtert, daß Arbeitsversionen von Programmen auf Diskette speicherbar sind. Durch Laden des Programms in das Anwendersystem kann die Arbeit sofort mit dem letzten Stand fortgesetzt werden.

Funktion der Kopplung

Die Kopplung benötigt das Betriebssystem ISM /10/ und ein spezielles Kommunikationsprogramm auf dem Entwicklungssystem. Der Monitor und das Kommunikationsprogramm sind in zwei Varianten, für parallele oder serielle Kopplung, verfügbar.
Die *parallele Kopplung* arbeitet auf der 16-Bit-Seite mit einem PPI-Schaltkreis und auf der 8-Bit-Seite mit einer geringfügig modifizierten ADA-Karte K 6020. Der Datenaustausch erfolgt mit Handshake. Die Übertragungsgeschwindigkeit beträgt etwa 1,2 kByte/s.
Die *serielle Kopplung* nutzt einen USART und einen Intervall-Timer auf der 16-Bit-Seite und eine V.24-Anschluß-

Anwender-system
PIT
USART
PROM
RAM
CPU
ZVE
Terminal

Bild 1 Beispiel einer Minimalkonfiguration für die Monitorarbeit

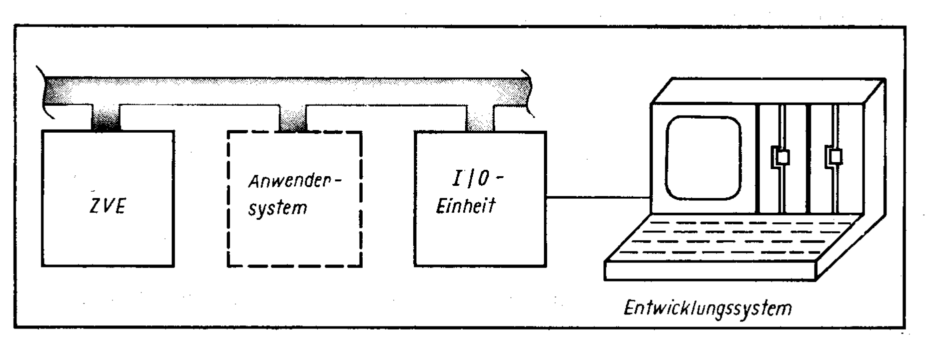


Bild 2 Enwicklungsplatz mit Kopplung zwischen 16-Bit-System und 8-Bit-Entwicklungssystem

steuerung ASV K 8021 auf der 8-Bit-Seite. Die Baudrate beträgt 9600 Bd. Die Handshakesignale wurden durch die Übertragung von Anforderungs- und Quittungszeichen für jedes gesendete Datum ersetzt. Damit kann auf zusätzliche Steuerleitungen verzichtet und die Kopplung auch mit einer Stromschleife aufgebaut werden. Die Datenübertragungsgeschwindigkeit beträgt etwa 300 Byte/s. Die Übertragung von 16 kByte benötigt in beiden Richtungen einschließlich Diskettenarbeit etwa 1,5 min.
Der Start des Monitors über die Kopplung erfolgt nach Aufruf des Kommunikationsprogramms und RESET des Monitors durch einen Anfangsdatenaustausch zur Synchronisation von Monitor und Kommunikationsprogramm.

Monitorfunktionen

Um eine Einschätzung der Leistungsfähigkeit des Monitors zu ermöglichen, werden seine Funktionen im Überblick dargestellt:

- Anzeigen, Ändern und Vergleichen von Speicherinhalten. Das Anzeigeformat ist wählbar.
- Suchen nach vorgegebenen Zeichenketten
- Rückübersetzung von Codebereichen
- Anzeige und Änderung der Register von CPU und Arithmetikerweiterung
- Anzeige und Beschreiben von E/A-Ports
- Programmabarbeitung unter Monitorkontrolle mit Adreßunterbrechungspunkten (Lesen des Operationscodes von einer angegebenen Adresse) und Datenunterbrechungspunkten (Beschreiben eines vorgebbaren Speicherbereiches)
- Laden von Systemprogrammen in den Speicher des 16-Bit-Systems von einer direkt am 16-Bit-System angeschlossenen Diskettenstation.

Während diese Funktionen auch ohne Kopplung zum Entwicklungssystem ausgeführt werden können, ist sie bei den nachfolgenden Funktionen erforderlich.

- Übertragen ausführbarer Programme vom 8-Bit-System zum 16-Bit-System
- Abbilden eines Speicherbereiches des 16-Bit-Systems auf ein Diskettenfile
- Routinen des Entwicklungssystems für Diskettenfileverwaltung; Bildschirm- und Tastaturbedienung können vom Monitor und von Anwenderprogrammen genutzt werden.

Neben der Kopplung besitzt der Monitor weitere Vorteile, auf die im folgenden hingewiesen werden soll.
Der Befehlsaufbau ist zeilenorientiert. Tafel 1 enthält die vollständige Befehlsliste des Monitors. Eine Befehlszeile

Tafel 1 Liste der Monitorbefehle

Go	Startet Programmausführung ab angegebener Adresse mit max. 4 Unterbrechungspunkten
N	Einzelschrittbetrieb, zeigt einen Befehl rückübersetzt an und führt ihn beim nächsten Schritt aus
X	Anzeige und Änderung der Register von CPU und Arithmetikerweiterung
D	Anzeige eines Speicherbereiches auf dem Bildschirm mit wählbarem Format (Byte und ASCII, Word, Word Integer, Short Integer, Long Integer, BCD, Short Real, Long Real, Temporary Real, Mnemonik)
S	Speicherinhalt ändern
M	Speicherbereiche verschieben
F	Suchen nach einer Konstanten in einem Speicherbereich
C	Vergleich zweier Speicherbereiche
I	Eingabe von einem Port und Anzeige des Wertes
O	Ausgabe zu einem Port
P	Ausgabe von ASCII-Zeichen zum Bildschrim
*	Nach * bis Zeilenende steht Kommentar
B	Laden ausführbarer Programme von direkt am 16-Bit-System angeschlossener Diskettenstation
Befehle für angeschlossenes Entwicklungssystem:	
L	Laden ausführbarer Programme vom Entwicklungssystem in den Speicher des 16-Bit-Systems
R	Wie L und zusätzlich Starten des Programms
T	Bildet einen 16-Bit-Speicherbereich auf einen Diskettenfile ab
E	Rücksprung zum Betriebssystem des Entwicklungssystems

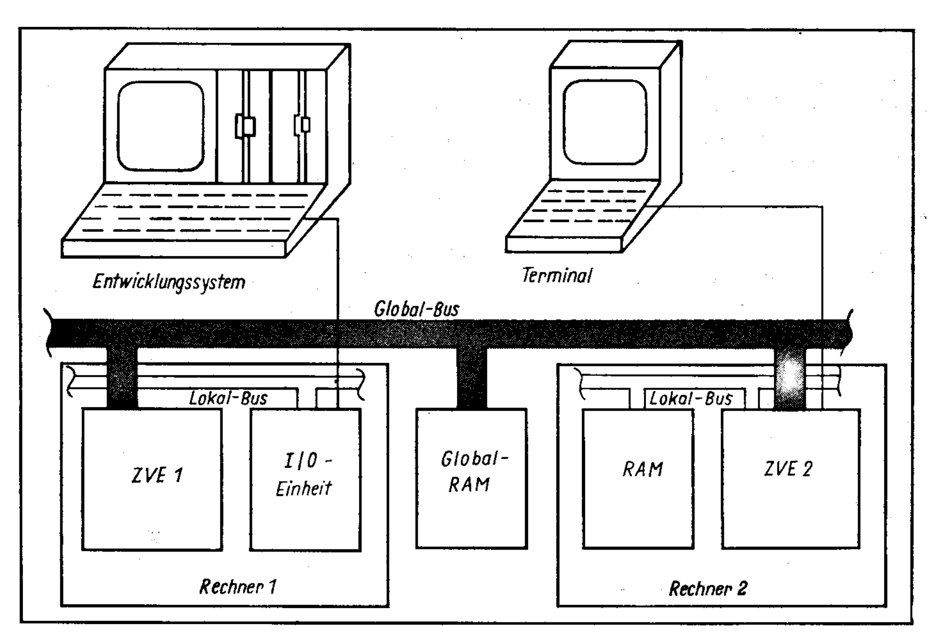


Bild 3 Entwicklungsplatz für ein Zweirechnersystem

kann mehrere Befehle enthalten. Außerdem können Wiederholungsfaktoren mit mehreren Klammerebenen angegeben werden. Beispielsweise bewirkt die Progrommzeile

5 < 12 < G, CS : 3B7 > ;D DS : 4A >

mit den Wiederholungsfaktoren 5 und 12, daß die Programmabarbeitung 12mal bis zur Adresse 3B7 des Codesegmentes ausgeführt und anschließend das Byte mit der Adresse 4A im Datensegment angezeigt wird. Diese Befehlsfolge führt der Monitor 5mal hintereinander aus. Fortsetzungsfaktoren bewirken, daß ein Befehl mehrfach mit verschiedenen Operanden ausgeführt wird. Der Befehl

20D 1000 : 0

zeigt 20 aufeinanderfolgende Speicherplätze ab 1000 : 0 an. Mit diesen Möglichkeiten lassen sich Programmschleifen und Programme, bei denen Interrupts auftreten, günstig testen. Eine wesentliche Arbeitserleichterung, besonders bei der komplexen Befehlsstruktur der I16–CPU, ist der eingebaute Rückübersetzer, der im Einzelschrittbetrieb die aktuelle Befehlszeile ausgibt.

In der Anfangsphase der Systementwicklung ist es vorteilhaft, daß der Einzelschrittbetrieb ohne zusätzliche Baugruppen realisiert wird. Im Einzelschrittbetrieb arbeitet der Monitor mit einem Software-Interrupt (INT 3).

Mehrrechnerarbeit

Der Monitor wurde über seine ursprüngliche Anwendung hinaus zur Programmentwicklung für Mehrrechnersysteme eingesetzt. Der Programmtest erfolgte auf einem Rechnersystem, das aus zwei Rechnern jeweils mit lokalem Bus und lokalen Ressourcen (RAM, PROM, I/O-Ports) und einem gemeinsamen (globalen) Speicher besteht (Bild 3). Über den globalen Speicher können Daten zwischen den Rechnern ausgetauscht werden. Der Monitor besitzt dafür jedoch keine Routine. Wird der Monitor aber auf beiden Rechnern installiert, dann können Daten über den globalen Speicher mittels Move-Befehl transportiert werden.
Diskettenarbeit ist nur über den Rechner 1 möglich. Programme für Rechner 2 werden zunächst in den Rechner 1 geladen und dann über den globalen Speicher zum Rechner 2 übertragen. Beim Speichern von Programmen ist der Vorgang umgekehrt. Durch diese Vorgehensweise wurde der Monitor auch für Mehrrechnerarbeit nutzbar.

✉ KONTAKT ☏
Zentralinstitut für Kybernetik und Informationsprozesse der AdW der DDR, Kurstraße 33; Berlin, 1086,
Tel. Dresden 4 63 32 15, Dr. Despang

Literatur
/1/ iRMX 86 Debugger Reference Manual, INTEL, 1981
/2/ Intel Development Systems Handbook 1985
/3/ 100-MHz-Logikanalysator mit IBM-PC. electronic-industrie 17 (1986) 5, 176
/4/ INTEL-Entwicklung auf ihrem PC. electronic-industrie 17 (1986) 5, 179
/5/ Bewährte Tools für neue hosts. electronic-industrie 17 (1986) 3, 66–70
/6/ Eine Personal Workstation für jeden Entwickler. mini micro magazin (1986) 3, 24–38
/7/ Klühe, B.: Mehrmikrorechner-Kommunikationssoftware MKS. Erscheint in Mikroprozessortechnik.
/8/ User's Guide for the iSBC 957B iAPX 86,88 Interface and Execution Package, INTEL
/9/ Funktionsbeschreibung NZVE I16–02. VEB Numerik Karl-Marx-Stadt 1984
/10/ Paschke, H.; Viertel, F.: Nutzerdokumentation Betriebssystem ISM.IEA, Berlin 1985

Personalcomputer in der Meßtechnik

Dr. Berndt Götze, Dr. Karl-Heinz Meusel
Technische Universität Dresden
Sektion Informationstechnik

Die beschleunigte Entwicklung und der breite Einsatz mikroelektronischer Schaltkreise sind mit völlig neuen Merkmalen der elektronischen Erzeugnisse verbunden. Das sind u. a.:
– Das Erhöhen der strukturellen und funktionellen Komplexität, wie es z. B. in der Entwicklung vom 4-Bit- bis zum 32-Bit-Mikroprozessor sehr deutlich wird.
– Der Übergang von der analogen zur digitalen Informationsverarbeitung und -übertragung, der sich z. B. bei den Kommunikationsdiensten vollzieht.
– Die enge Wechselwirkung von Hard- und Softwarekomponenten bei steigenden Anforderungen an die Komplexität der Software.
– Das Erhöhen der Typenvielfalt hoch- und höchstintegrierter Schaltkreise durch die in zunehmendem Maße bereits bei kleinen Stückzahlen ökonomischen kundenspezifischen Schaltkreise.
– Das Erhöhen der Arbeitsgeschwindigkeit elektronischer Geräte durch neue Technologien und Steigern des Integrationsgrades der mikroelektronischen Bauelemente.
Diese im Vorspann genannten qualitativ neuen Merkmale stellen naturgemäß höhere Ansprüche an die Entwicklungs-, Test- und Prüfmittel des Ingenieurs. Insbesondere steigen die Anforderungen an solche digitalen Meßgeräte wie Logikanalysatoren. Hier kommt es darauf an, eine Vielzahl korrelierender Signale aufzuzeichnen und auszuwerten. Die außerordentlich hohe logisch-funktionelle Komplexität dieser Signalströme erfordert einerseits leistungsfähige datenabhängige Selektionsmechanismen bei der Signalaufzeichnung und andererseits problemorientierte und gegebenenfalls automatisierbare Auswertestrategien. Um eine solche Meßtechnik effektiv einsetzen zu können, ist dem Nutzer eine weitgehende Unterstützung beim Gerätebedienen und Auswerten der Meßinformationen zu geben. Diese Forderungen sind nur über eine entsprechende Rechnerleistung zu erfüllen.
Der Stand der Mikrocomputertechnik rechtfertigt es, diese Rechnerleistung dem Meßgerät bzw. dem Gerätesystem unmittelbar zuzuordnen oder sie direkt zu integrieren. Das ist eine Folge der drastischen Miniaturisierung und erheblichen Kostensenkung bei gleichzeitigem Vervielfachen der Leistungsfähigkeit. Beispiele für derartig „intelligente" Meßgeräte sind in /1, 2, 7, 8/ zu finden.
Auf dem Gebiet der Logikanalyse hat sich bezüglich der „Geräteintelligenz" eine äußerst rasche Entwicklung vollzogen. Aber auch auf anderen Gebieten der Meß- und Analysetechnik vollzieht sich ein derartiger Wandel.
Am Beispiel der Anforderungen der Logikanalyse werden prinzipielle Möglichkeiten zur Implementierung der Geräteintelligenz diskutiert.

1. Lösungsmöglichkeiten für die Geräteintelligenz

Um elektronische Meßgeräte unmittelbar mit einer bestimmten Rechnerleistung auszurüsten, bieten sich die in Bild 1 dargestellten grundsätzlichen Möglichkeiten an.

1.1. Integration von Mikrorechnern in Meßgeräte

Um kompakte tragbare Meßgeräte mit Rechnerleistung auszurüsten, werden die wichtigsten Komponenten von Mikrorechnern (Zentrale Verarbeitungseinheit, Programm- und Datenspeicher, Kommunikationsmittel unterschiedlicher Leistungsfähigkeit) in das Gefäßsystem der Meßgeräte integriert. Diese Vorgehensweise ist nahezu in allen modernen Logikanalysatoren, in mikrorechnergesteuerten Oszilloskopen /8/ und ähnlichen Geräten anzutreffen. In jedem Falle ist es dazu erforderlich, spezielle Mikrorechnerbaugruppen zu entwickeln, die dem Einsatzfall optimal angepaßt sind. Gut geeignet sind für solche Zwecke Einchipmikrorechner, insbesondere für weniger komplexe Meßgeräte, die nicht so hohe Anforderungen an den Softwareumfang stellen. In der Regel werden Einschränkungen, bezogen auf Speicherumfang und rechentechnische Peripherie vorgenommen. Das ist unter Umständen mit einer Einschränkung an Flexibilität verbunden.

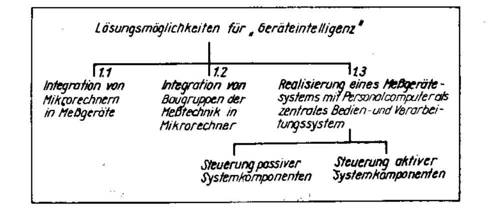


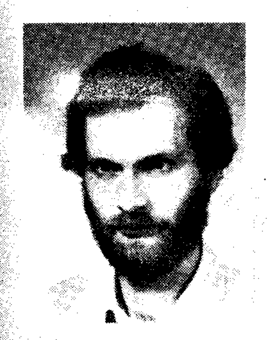
Bild 1 Lösungsmöglichkeiten für die Geräteintelligenz

1.2. Integration von Baugruppen der Meßtechnik in Mikrorechner

Diese Lösungsmöglichkeit liegt dem mikrorechnergesteuerten Logikanalysator LA 32/20 /1, 2, 3/ und dem Logikgenerator LG 32/20 /4/ zugrunde. In /1/ wurden ausführlich die Vorteile dieser Variante diskutiert. Gegenwärtig ist in dieser Richtung ein Trend zu beobachten, durch Zusatzbaugruppen, sogenannte *Addin-Baugruppen* /5, 6, 7, 9/, Personalcomputer (PC) mit einer geeigneten meßtechnischen Peripherie auszurüsten. Damit entstehen neue Einsatzgebiete für die in großen Stückzahlen produzierten PC einerseits und kostengünstige Lösungen für den Anwender ande-

rerseits. Für den Einsatz von Personalcomputern als Intelligenz von Meßgeräten zeichnen sich u. a. folgende Vorteile ab:

- Der PC wird in hohen Stückzahlen bei relativ niedrigen Kosten produziert und kann in wachsendem Maße in nahezu allen volkswirtschaftlichen Bereichen als vorhanden und oftmals als nicht ausgelastet vorausgesetzt werden, d. h. er stellt unter diesen Voraussetzungen keinen zusätzlichen Kostenaufwand für ein neues Einsatzgebiet dar.
- PC verfügen prinzipiell über alle notwendigen rechentechnischen Mittel wie leistungsfähige Verarbeitungseinheit, große Arbeitsspeicher, Massenspeicher, Kommunikationsmittel (Grafikdisplay und alphanumerische Tastatur), Betriebssysteme und verschiedenste Basissoftware.
- PC bieten damit neben dem Einsatz in der Meßtechnik auch weiterhin alle Möglichkeiten zur Programmentwicklung, Textverarbeitung, rechnergestützten Ingenieurarbeit usw. und sind damit am Arbeitsplatz des Elektronikingenieurs multivalent nutzbar.
- Gegenüber bisherigen Lösungen für mikrorechnergesteuerte Meßtechnik können mit dem Einsatz von PC auch qualitativ neue Eigenschaften realisiert werden, wie die Integration in lokale Netze, die Steuerung ganzer Gerätesysteme bis hin zur Realisierung kleiner Expertensysteme für meßtechnische Probleme, d. h. bis hin zum Einführen von Methoden der künstlichen Intelligenz. Das hängt jedoch in starkem Maße von der Leistungsfähigkeit des eingesetzten PC ab.





Berndt Götze (32) studierte von 1974 bis 1978 an der Sektion Informationstechnik der TU Dresden. 1982 promovierte er mit einer Arbeit zu Multiprozessorsystemen zum Dr.-Ing. Er bearbeitet an der Sektion Informationstechnik als wissenschaftlicher Oberassistent die Gebiete Software und Logikanalyse.

Karl-Heinz Meusel (35) studierte ebenfalls Informationstechnik an der TU Dresden und promovierte 1979 auf dem Gebiet der Rechentechnik zum Dr.-Ing. Tätig als wissenschaftlicher Oberassistent an der Sektion Informationstechnik arbeitet Dr. Meusel an wissenschaftlichen Aufgaben in der Mikroprozessortechnik und Logikanalyse.

1.3. PC in Meßsystemen

Die physikalische und funktionelle Integration von Baugruppen der Meßtechnik in PC unterliegt jedoch Einschränkungen bereits durch die Aufnahmefähigkeit der Gefäßsysteme. Auswege bieten hier Buserweiterungen und Zusatzgefäßsysteme /6/ für PC und das volumenmäßige Reduzieren der Zusatzbaugruppen mittels Erhöhen des Integrationsgrades der dort eingesetzten Bauelemente. Bezogen auf einen realen technologischen Stand wird es jedoch immer eine Grenze für den implementierbaren Funktionsumfang geben. Diese Anforderungen erzwingen konfigurierbare, d. h. an die Meßaufgaben anpaßbare Systeme, die nicht mehr in vorhandene Mikrorechner integrierbar sind, sogenannte *Add-on-Baugruppen*. Für den Einsatz von PC in solchen größeren Meßgerätesystemen bieten sich zwei grundsätzliche Wege an. Der erste Weg besteht darin, daß der PC über die bereits erwähnte Buserweiterung passive Systemkomponenten steuert. Passive Systemkomponenten sind dabei solche, die über keine eigene Rechnerleistung verfügen. In diesem Falle besteht sozusagen eine funktionelle Identität zu 1.2., d. h. zur Integration von peripheren Meßgerätebaugruppen in Mikrorechnern. Der zweite Weg besteht darin, aktive Systemkomponenten zu steuern, d. h. solche, die über eigene Rechnerleistung verfügen, die entsprechend 1.1. in die Gerätekomponente integriert wurde. In diesem Fall übernimmt der PC die Aufgabe der koordinierenden Steuerung der Systemkomponenten, der Mensch-Maschine-Kommunikation und des Auswertens der Meßinformationen in einem hierarchisch angeordneten Gerätesystem. Die aktiven Systemkomponenten realisieren die eigentlichen meßtechnischen Aufgaben und sind in der Lage, komplexe Anweisungen des PC auszuführen und Meßinformationen vorzuverarbeiten. Die Leistungsfähigkeit eines solchen Systems steigt also gegenüber dem ersten Weg deutlich an. Als weiterer Vorteil ergibt sich, daß für eine Vielzahl unterschiedlicher Geräte über den PC eine gleichartige Bedienstrategie realisierbar ist.

2. PC zur Meßgerätesteuerung

Soll ein PC die Steuerung von Meßgerätekomponenten übernehmen, bieten sich prinzipiell die Lösungsansätze nach 1.2. oder 1.3. an. Bei der Integration von Meßgerätehardware in den PC muß davon ausgegangen werden, daß der PC-Hersteller schon bei der Entwicklung von Gefäßsystem, Stromversorgung und Adressierbarkeit eine umfang-

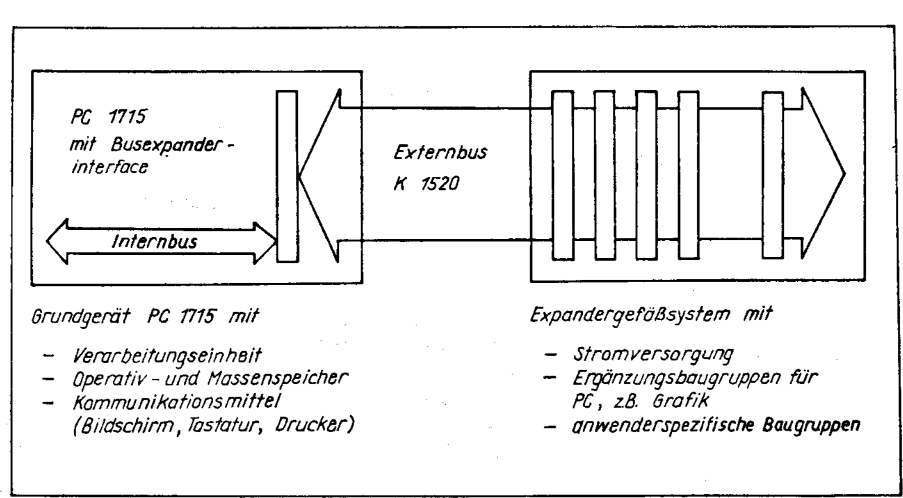
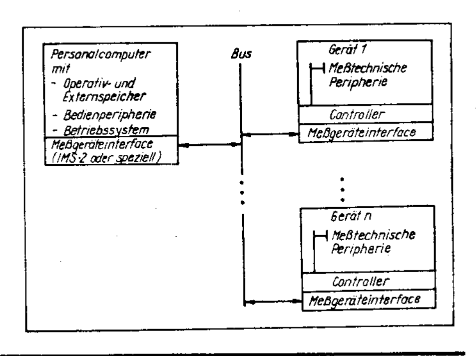


Bild 2 Meßgerätesteuerung mit PC 1715

Bild 3 Steuerung aktiver Meßgerätekomponenten



reiche Erweiterbarkeit vorausgesetzt hat. Das schließt das nachträgliche Konzipieren solcher Einsatzfälle nach Lösungsansatz 1.2. aus. Es kann nur die Steuerung einer meßtechnischen Peripherie, die in einem Expandergefäß untergebracht ist, in Frage kommen.
Da in der DDR in allen Bereichen der Volkswirtschaft in großem Umfang der PC 1715 zum Einsatz kommt, soll seine Verwendung als zentrale rechentechnische Komponente eines Meßgerätes diskutiert werden.
Aus der Analyse der Grundausstattung dieses PC ist ersichtlich, daß die Grafikfähigkeit und eine Schnittstelle zum Ansteuern passiver Baugruppen fehlen. Ein Lösungsansatz kann darin bestehen, daß der verfügbare freie Steckplatz genutzt wird, um ein spezielles Interface einzubauen. Über dieses werden sowohl die Vollgrafikbaugruppe als auch weitere meßtechnische Peripherie, die sich gemeinsam im Expandergefäß befinden, an den PC 1715 angeschlossen. Dabei wird der PC 1715 in seinem normalen Einsatzspektrum nicht eingeschränkt. Mit der Austauschbarkeit der meßtechnischen Peripherie im Expandergefäß (z. B. Logikanalysator, Logikgenerator, IMS2-Bus) ergeben sich somit eine Vielzahl weiterer Einsatzmöglichkeiten. Die Prinziplösung ist in Bild 2 dargestellt.
Um auch bei der meßtechnischen Peripherie vorhandene Entwicklungen nutzen zu können /1, 2, 3, 4/, bietet es sich an, als Interface in den PC 1715 eine Buserweiterung mit einem K-1520-Bussystem einzubauen. Das bietet darüber hinaus auch den Vorteil, alle verfügbaren K-1520-OEM-Baugruppen einsetzen zu können. Damit sind Anwendungen in der Prozeßautomatisierung, Prozeß-, Steuerungs- und Regelungstechnik, der Medizintechnik und in vielen anderen Gebieten möglich. Die Adreßräume des Speichers und der Peripherie werden so aufgeteilt, daß auf die im Expandergefäß untergebrachten Komponenten zugegriffen werden kann, ohne daß Konflikte mit den Hardwarekomponenten des PC 1715 auftreten.

3. PC in Meßsystemen

In der Zukunft werden nicht nur einzelne Meßgeräte, sondern auch mehrere Meßgeräte im Komplex an der Lösung einer Meßaufgabe beteiligt sein. Deshalb ist es auch erforderlich, sie gemeinsam zu steuern und zu überwachen.
Diese zentrale Funktion in Meßsystemen kann ebenfalls ein PC übernehmen. Jedoch ist es dann nicht mehr möglich, daß er alle Systemkomponenten direkt ansteuern kann. Hier ist es sinnvoll, den Personalcomputer durch ein meßtechnisches Interface aufzurüsten.
Ein solches meßtechnisches Interface setzt dann voraus, daß sowohl der Anschluß im PC als auch die meßtechnischen Komponenten mit einer minimalen rechentechnischen Verarbeitungsleistung ausgestattet sein müssen, um Datenübertragungsprotokolle und Steuerungsfunktionen realisieren zu können (siehe Bild 3). Die meßtechnischen Komponenten können entsprechend Punkt 1.1. oder 1.2. mit der notwendigen Rechnerleistung ausgestattet werden.
Diese Art des Anschlusses entbindet den PC von der direkten Gerätesteuerung, der dann nur noch Datenmassive mit logischen Bezügen bearbeiten muß. Das Zuordnen dieser Daten zur konkreten gerätetechnischen Peripherie übernimmt die in der meßtechnischen Komponente installierte Verarbeitungsleistung.
Ein solches Konzept ist mit dem PC 1715, wegen der fehlenden Grafikfähigkeit, zu geringer Verarbeitungsleistung und zu kleinem Speichervolumen, nicht mehr effektiv realisierbar.

Literatur

/1/ Götze, B.; Meusel, K.-H.; Nicklisch, G.: Mikrorechnergesteuerter Logikanalysator LA 32/20. Wissenschaftliche Zeitschrift der TU Dresden. 33 (1984) 4, S. 253–259
/2/ Götze, B.; Meusel, K.-H.; Nicklisch, G.: Logikanalysator LA 32/20. radio fernsehen elektronik. Berlin 34 (1985) 10, S. 626–629 und 11 S. 719–722
/3/ Meusel, K.-H.; Piepiorra, F.: Softwareanalyse mit Logikanalysesystem LAS 20. radio fernsehen elektronik. Berlin 35 (1986) 2, S. 71–74
/4/ Schnetter, V.; Naumann, R.; Frühauf, U.: Logikgenerator-Bestandteil eines digitalen Testsystems. Nachrichtentechnik/Elektronik. Berlin 35 (1985) 8, S. 301–302
/5/ Dolch, V.: Personalcomputer verändern die Meßtechnik. VDI Nachrichten Düsseldorf (1985) 12 Nr. 52
/6/ Meßwerterfassung und -verarbeitung: Aktuelles und Trends. Elektronik München (1986) 4, S. 108–110
/7/ Vieten, M.: Erfassung schneller Meßsignale mit dem PC. Elektronik München (1986) 4, S. 113–117
/8/ Rampey, F.; Watry, B.; Austgen, P.: Microcomputer statt Drehknopf: Eine neue Oszilloskopgeneration. Elektronik München (1980) 19, S. 53–56
/9/ Lang, K.: Personalcomputer-Intelligenz als zentrale Meßeinheit. VDI-Nachrichten Düsseldorf Nr. 40/3. 10. 86

Kleincomputer KC 85/3

Hardwarekonzept

Dr. Werner Domschke
VEB Mikroelektronik „Wilhelm Pieck“ Mühlhausen

In konsequenter Fortführung des Kleincomputersystems KC 85 des VEB Mikroelektronik „Wilhelm Pieck“ Mühlhausen wurde nach dem KC 85/2 der KC 85/3 entwickelt. Bei diesem Kleincomputer wird das modulare Konzept sowohl in der Gerätetechnik (Hardware) als auch in der Programmtechnik (Software) beibehalten, so daß sowohl die Zusatzbaugruppen, z. B. die Module, als auch die Programme des KC 85/2 bei dem KC 85/3 weiterverwendet werden können. Der wesentliche Unterschied beider Kleincomputer liegt im Gebrauchswert. Bei dem KC 85/3 ist der BASIC-Interpreter im System-ROM enthalten. Dadurch stehen dem Anwender in der Grundausrüstung neben dem etwa dreifachen Programmspeicher für BASIC-Programme eine Reihe von zusätzlichen BASIC- und Betriebssystemanweisungen und -unterprogrammen zur Verfügung.
Das Kleincomputersystem KC 85 wurde in erster Linie für den Einsatz in der Lehre, Aus- und Weiterbildung, in Forschung und Entwicklung, in Büros und für andere Einsatzfälle konzipiert, bei denen das Lernen und die Verarbeitung von geringen Datenmengen im Vordergrund steht.
Zum Lieferumfang des KC 85/3 gehört eine umfangreiche Dokumentation, bestehend aus:
- Gerätebeschreibung
- Systembeschreibung
- BASIC-Handbuch.

Diese Unterlagen enthalten detaillierte Angaben, die zum Betrieb des KC 85/3 erforderlich sind.
Im folgenden soll das Hardwarekonzept vorgestellt werden. Das Softwarekonzept wird ausführlich in MP 3/87 erläutert.

Charakteristik

Der KC 85/3 besteht aus dem Grundgerät (BASIS DEVICE) und der Tastatur (KEYBOARD), die mit dem Gerät durch eine dünne flexible Leitung verbunden ist.
Zum Betrieb des KC ist der Anschluß eines Datensichtgerätes erforderlich. Das kann ein Schwarzweiß- oder Farbfernsehgerät oder ein Monitor sein. Sollen erarbeitete Programme oder Daten dauerhaft gespeichert werden, so ist ein handelsüblicher Kassettenrecorder zu verwenden.
Die Bilder können auf dem Fernsehgerät in 16 Vordergrund- und 8 Hintergrundfarben vollgrafisch dargestellt werden. Die Erzeugung von Tönen über den eingebauten Generator ist zweikanalig mit je 5 Oktaven möglich. Die

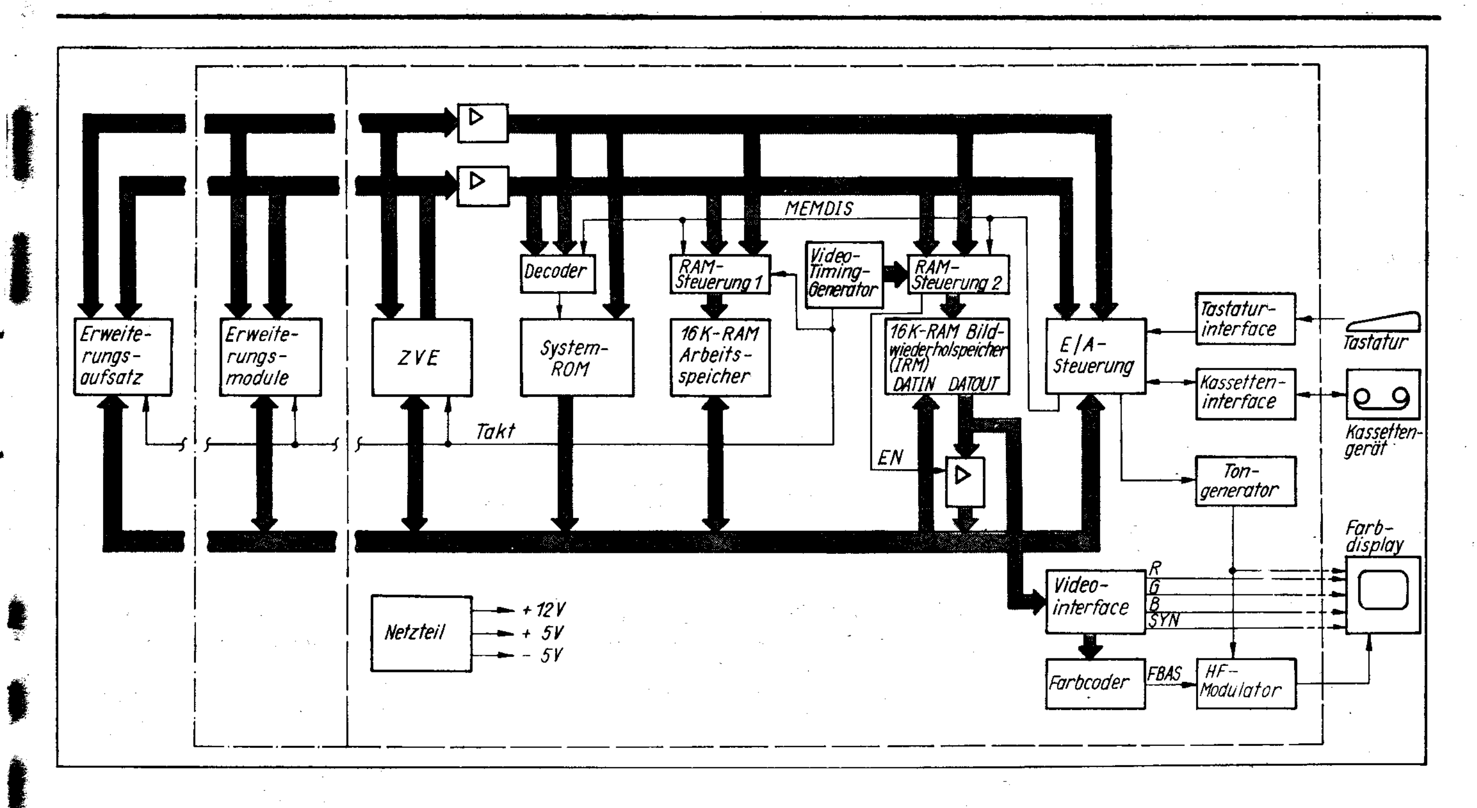


Bild 1
Blockschaltbild

technischen Daten sind in Tafel 1 zusammengefaßt.

Baugruppen

Im Bild 1 ist das Blockschaltbild dargestellt. Die Baugruppen sollen im folgenden kurz erläutert werden.
Die ZVE enthält den Mikroprozessor UB 880 D und die RESET-Logik. Diese Logik ermöglicht es, den System-ROM am oberen Ende des Adreßraumes des Prozessors anzuordnen. Da bei dem KC 85/3 zwei RESET-Quellen vorhanden sind, realisiert die RESET-Logik unterschiedliche Reaktionen. Bei Netzspannungseinschaltung wird der gesamte Speicher gelöscht, die Systemarbeitszellen und die internen Peripherieschaltkreise initialisiert und das Anfangs-Menübild auf dem Fernsehgerät dargestellt. Bei Betätigung der RESET-Taste werden die Systemarbeitszellen und die internen Peripherieschaltkreise initialisiert und das Menübild dargestellt. Somit bleiben Anwenderprogramme und Daten erhalten.
Der *System-ROM* besteht aus zwei Schaltkreisen zu je 8 KByte. Im ersten ROM ist der Kern des BASIC-Interpreters und im zweiten ROM die BASIC-Ergänzungen, das Betriebssystem HC-CAOS und die Zeichenbildtabellen für Groß- und Kleinbuchstaben gespeichert. Beide Schaltkreise sind mit MEMDIS-Signalen inaktiv zu schalten. Dann ist der jeweilige Adreßbereich frei für andere Speicher, die z. B. die Interpreter für andere Programmiersprachen oder ein anderes Betriebssystem enthalten können. Der *RAM* hat eine Größe von 16 KByte und belegt den Adreßbereich 0 ... 3FFFH. Etwa 0,2 KByte werden vom Betriebssystem für Arbeitszellen im Bereich 14 0H ... 1 FFH nach jedem RESET initialisiert. Der Anwender kann diese Arbeitszellen aber an beliebiger Stelle im RAM anordnen. Die restlichen 15,8 KByte sind ohne Einschränkung vom Anwender nutzbar.
Der RAM kann schreibgeschützt und vom Prozessorbus offline geschaltet werden. In allen Zuständen ist der Datenerhalt durch Refresh gewährleistet.
Der *Bildwiederholspeicher (IRM)* ist ein Zweitorspeicher, auf den zeitlich verschachtelt der Prozessor schreibend und lesend und das Video-Interface (VIF) nur lesend zugreifen. Im IRM sind die auf dem Bildschirm sichtbaren Informationen als Bildpunkte (Pixel-RAM), als Farbinformation (Color-RAM) und als ASCII-Zeichen (Video-RAM) gespeichert. Das Video-Interface greift nur auf die Informationen im Pixel- und Color-RAM zu und wandelt diese in normgerechte Fernsehsignale um. Der IRM enthält weiterhin die meisten System-Arbeitszellen, den Kassettenpuffer, den Speicher für die vom Anwender frei definierbaren Funktionstasten, den Speicher für die Bildausschnitts- (Fenster-) Vektoren, einen Bereich für Peripherie-Verbindungsprogramme und einen vom Anwender nutzbaren freien Speicher.
Jeweils 32 Bildpunkten (8 Byte des Pixel-RAM) ist eine Farbtabelle (1 Byte im Color-RAM) zugeordnet. Damit läßt sich für dieses Feld eine von 8 Hintergrundfarben (Bit im Pixel-RAM ist Null) und eine von 16 Vordergrundfarben (Bit im Pixel-RAM ist Eins) und die Blinkfunktion für die Vordergrundfarben auswählen.
Der IRM kann von Prozessorbus offline geschaltet werden. Der Datenerhalt und die Bilddarstellung sind dabei gewährleistet.
Der *Video-Timing-Generator (VTG)* wird durch einen Quarz mit der doppelten PAL-Farbträgerfrequenz von etwa 8,866 MHz gesteuert. Über Teilerstufen werden die Prozessortaktfrequenz von etwa 1,75 MHz, die PAL-Trägerfrequenz von etwa 4,433 MHz, die Zeilen- und Bildsynchronsignale für das Fernsehgerät von etwa 15,6 KHz bzw. 50 Hz, die Speichersteuersignale der dynamischen Speicher und die Adreßsignale für den IRM erzeugt. Damit passen sich alle Baugruppen des KC 85/3 in ein Taktregime ein.
Das *Video-Interface (VIF)* liest die Bildinformationen aus dem Pixel- und Color-RAM des IRM aus, serialisiert sie und paßt die Pegel der Rot-, Grün- und Blauinformationen (RGB) an die genormten Werte von 0,75 V an 75 Ohm an. An dieser Anschlußstelle, die an der Rückseite des KC zugängig ist, kann der Farbmonitor mit RGB-Buchse oder der Farbfernsehempfänger mit PERI- oder SCART-Buchse betrieben werden. Mit diesem Anschluß erhält man die höchste Bildqualität, da alle Farbinformationen mit einer Bandbreite von etwa 7,5 MHz auf dem Bildschirm darstellbar sind.
Der *Farbcoder* setzt die RGB-Signale in ein Fernsehsignal um, das der PAL-Norm nahekommt. Das läßt eine kostengünstige Realisierung des Farbcoders zu. Am Ausgang des Farbcoders steht das FBAS-Signal-Gemisch zur Verfügung (Farb-Bild-Austast-Syn-

Tafel 1 Technische Daten des KC 85/3

Hersteller	VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen im VEB Kombinat Mikroelektronik
Bauform	Grundgerät mit abgesetzter Tastatur
Abmessungen	Grundgerät 385 × 250 × 77 mm³ Tastatur 296 × 152 × 18/29 mm³
Masse	etwa 4700 g
Grundgerät	
Betriebsspannung	220 V
Leistungsaufnahme	etwa 25 W
Prozessortyp	UB 880 D
Schreib-/Lesespeicher	16 × U 256 C (32 KByte)
Festwertspeicher	2 × U 2364 D (8 KByte)
für Anwender freier RAM	etwa 17 KByte
Bildwiederholspeicher	13,75 KByte
Bildaufbau	vollgrafisch 320 × 256 Bildpunkte
Nutzbare Bildpunktzahl	81 920
Zeichendarstellung	8 × 8 Bildpunkte
Anzahl der Zeichen/Zeile	40
Anzahl der Zeilen	32
Vordergrundfarben	16 (einschl. Schwarz und Weiß)
Hintergrundfarben	8 (einschl. Schwarz und Weiß)
Betriebssystem	HC-CAOS V3.1 (4,5 KByte)
Programmiersprache im ROM	BASIC (10,5 KByte)
Zeichenbildtabellen	2 (1 KByte)
im ROM enthaltenen Zeichenbilder	128
vom Anwender definierbare Zeichenbilder	beliebig viele
Zeichengenerator	mittels Software
Bildschirmeinteilung	durch Fenster (windows)
gleichzeitig definierbare Fenster	10
Anzeigeeinheit	handelsübliches Farb- oder Schwarzweißfernsehgerät, Farbmonitor oder Fernbildschreiber
Anschlußmöglichkeiten der Anzeigeeinheit	Antennenanschluß VHF Band III Kanal 8, FBAS-Anschluß, RGB-Anschluß (kompatibel zur SCART-Norm)
Farbfernsehnorm	PAL
Tongeneratoren	2
Tonumfang	5 Oktaven
Tonausgang	Diodenbuchse 2kanalig konstante Pegel, RGB-Ausgang 1kanalig mit Lautstärkeeinstellung
Lautstärkeumfang	32 Stufen
Anzahl der BASIC-Anweisungen	112
davon mathematische Funktionen	11
davon Stringfunktionen	10
mathematische Operationen	13
vom Anwender nutzbare Systemunterprogramme	77
Tastatur	frei beweglich, Elastomerprinzip
Tastaturprozessor	U 807 D
Verbindungsleitung	einadrig abgeschirmt
Stromversorgung	über Verbindungsleitung
Anzahl der Tasten	64
Cursorsteuertasten	4
Editiertasten	4 (INS, DEL, CLR, HOME)
Programmsteuertasten	3 (BRK, STOP, ENTER)
alphanumerische Tasten	45
Umschalttasten	2 (SHIFT, SHIFT LOCK)
Funktionstasten	6 (zweifach belegbar)
Tastenanordnung	gemäß Schreibmaschine

Tafel 2 Module zum KC 85. Der Modul M006 BASIC ist für den KC 85/2 entwickelt worden. Zur Vollständigkeit wurde dieser Modul mit angegeben.

Bezeichnung/Funktion	aktiv/inaktiv	Modulsteuerung Segmentierung	Segmentgröße	Termin der Produktionseinführung (Quartal)
M001 DIGITAL IN/OUT 16 Kanäle digital Ein-Ausgabe	×			IV/87
M003 V.24 2 Vollduplexkanäle V.24	×			III/86
M005 USER Leermodul für eigene Schaltungen	vom Anwender aufzubauen			III/86
M006 BASIC (BASIC-Interpreter für KC 85/2)	×	×	1 × 16 K	II/86
M007 ADAPTER Busverlängerung für Inbetriebnahme M005				III/86
M010 ADU 1 4 Kanäle Analog-Eingabe	×			IV/87
M011 64 K RAM 64 KByte RAM-Erweiterung	×	×	4 × 16 K	IV/87
M012 TEXOR Textverarbeitung, Sortier- und Suchprogramm	×	×	1 × 8 K	II/87
M022 EXPANDER RAM 16 KByte RAM-Erweiterung	×	×	1 × 16 K	I/86
M025 USER PROM 8 K 8 KByte EPROM	×	×	1 × 8 K	I/86
M027 DEVELOPMENT Editor, Assembler, Temo, Reassembler	×	×	1 × 8 K	II/87
M026 FORTH FORTH-Interpreter	×	×	1 × 8 K	II/87

chron-Signal-Gemisch). Dort können alle Farbfernsehempfänger mit PAL-Decoder bzw. alle Schwarzweiß-Fernsehempfänger angeschlossen werden, die eine AV-Buchse enthalten. Die Bildqualität ist gegenüber dem RGB-Anschluß geringer, da wegen des Farbfernsehprinzips die Bandbreite des Farbsignals auf etwa 1 MHz begrenzt wird (Bandbreite Luminanzsignal: 7,5 MHz).
Im *HF-Modulator* wird das FBAS-Signal einem HF-Träger aufmoduliert. Dieses Signal kann mit handelsüblichen Fernsehempfängern im VHF-Band III Kanal 8 empfangen werden. Die Bildqualität ist gegenüber dem FBAS-Ausgang geringer, da die Video-Bandbreite auf etwa 5 MHz begrenzt wird.
Die Baugruppe *E/A-Steuerung* besitzt vielfältige Aufgaben. Sie steuert

- die internen Speicher mittels der MEMDIS-Signale
- das Tastaturinterface
- das Kassetteninterface
- die Tongeneratoren

und ermöglicht neben dem Video-Interface die Kommunikation mit dem Bediener bzw. schafft die Voraussetzungen zum Speichern von Programmen und Daten auf Magnetband.
Das *Tastaturinterface* wandelt die von der Tastatur kommenden Signale so um, daß sie von der E/A-Steuerung weiterverarbeitbar sind. Gleichzeitig ist hier die Stromversorgung der Tastatur realisiert.
Das *Kassetteninterface* wandelt die

Computersignale in Töne um, die von einem handelsüblichen Kassettenrecorder aufgezeichnet werden können bzw. wandelt die von einem Recorder aufgezeichneten Töne in Computersignale um. Somit ist die Handhabung des Recorders wie bei Musikaufnahmen, und es können übliche Kassetten verwendet werden. /1/

Der *Tongenerator* erzeugt Rechtecksignale unterschiedlicher Frequenz und unterschiedlicher Amplitude. In zwei Kanälen lassen sich Frequenzen von etwa 27 Hz bis etwa 20 kHz einstellen. Davon sind Töne in 5 Oktaven sinnvoll einzuordnen. Diese Signale stehen zweikanalig an der Diodenbuchse „TAPE“ mit konstantem Pegel zur Verfügung. Nach Mischung und 32stufiger Lautstärkeregelung kann das Tonsignal an dem RGB-Steckverbinder entnommen werden.

Das *Netzteil* erzeugt aus 220 V Netzwechselspannung die internen Versorgungsspannungen +12 V, +5 V, −5 V. Die Stromversorgung ist so ausgelegt, daß zusätzlich an den Computer zwei Module mit einer maximalen Stromaufnahme von je 100 mA (+12 V), 250 mA (+5 V) bzw. 5 mA (−5 V) angeschlossen werden können.

Die *Tastatur* ist am Grundgerät steckbar mit einer abgeschirmten Eindrahtleitung verbunden. Über diese Leitung wird die Stromversorgung vom Computer zur Tastatur- und die Signalübertragung von der Tastatur zum Computer verwirklicht. Die parallele Abfrage der 64 Tasten und die Serialisierung der Information übernimmt der CMOS-Tastaturprozessorschaltkreis U 807 D.

Für *Erweiterungsmodule* sind im Grundgerät zwei Steckplätze (Schächte) vorgesehen. Damit ist der Speicherbereich des KC 85 erweiterbar oder wird der Anschluß von Peripheriegeräten oder von informationserzeugenden bzw. -verarbeitenden Geräten ermöglicht. Zu diesem Zweck sind Speichermodule (RAM- und ROM-Module), Interfacemodule (z. B. V.24, digitale Ein-/Ausgabe-, analoge Eingabemodule) in Produktion bzw. in Vorbereitung.

An dem an der Rückseite des Grundgerätes herausgeführten Rechnerbus (expansion interface) können *Erweiterungsaufsätze* am KC 85 angeschlossen werden. Damit läßt sich die Einsatzvielfalt noch vergrößern.

Da der Rechnerbus nur für die internen Baugruppen verstärkt wird, muß bei der Entwicklung von Modulen oder Aufsätzen darauf geachtet werden, daß der Bus je Erweiterung mit maximal einer low-power-shottky-TTL-Last beaufschlagt wird.

Bild 2 Speicheraufteilung des KC 85/3 mit zwei Modulen M022

Erweiterungskonzept

Da das Grundgerät des KC 85/3 für den Einstieg in die Computertechnik vorgesehen ist, wurde bei der Entwicklung auf eine umfangreiche Ausbaufähigkeit Wert gelegt. Dem Benutzer ist es somit möglich, das Computersystem entsprechend seinen Anforderungen zu konfigurieren.

Das Grundprinzip besteht darin, daß mehrere Module auch vom gleichen Typ unabhängig vom Modulsteckplatz (im Grundgerät oder im Modulaufsatz) gleichzeitig betrieben werden können, ohne daß am jeweiligen Modul eine Hardwareänderung vorgenommen werden muß. Das wird erreicht, indem in jedem Modul eine sogenannte Modulsteuerung enthalten ist, die z. B. bei Speichermodulen folgende Aufgabe erfüllt:

- Der Modul kann aktiv sein (der Prozessor kann mit dem Modul arbeiten) oder inaktiv sein (der Modul ist vom Prozessorbus getrennt).
- Der im Modul enthaltene Speicher wird in Segmente zu maximal 16 KByte Umfang aufgeteilt.
- Jedem Speichersegment wird eine Anfangsadresse in der Schrittweite der Segmentgröße zugeordnet, d. h. das jeweilige Speichersegment ist im Adreßraum des Prozessors verschiebbar.
- Der Speicher kann schreibgeschützt werden, so daß RAM wie ROM arbeiten.

Die Programmierung der Modulsteuerung erfolgt vom laufenden Programm aus. Dadurch kann auch während der Programmabarbeitung der Arbeitsspeicher entsprechend der gerade ablaufenden Aufgabe konfiguriert werden /2/.

Die Modulsteuerung wird in analoger Form auch für Ein-/Ausgabemodule verwendet, so daß z. B. auch mehrere V.24-Schnittstellen quasi gleichzeitig betrieben werden können.

Am Beispiel des Moduls M022 EXPANDER RAM soll das Prinzip kurz erläutert werden.

Im Adreßraum des Grundgerätes ist eine Lücke von 4000 H bis 7 FFFH. In diesem Bereich kann ein M022 arbeiten. Dem Anwender stehen damit 32 KByte Arbeitsspeicher zur Verfügung. Wird ein weiterer M022 gesteckt, so muß dieser auf den Adreßbereich 800 H bis 0BFFFH programmiert werden. Damit steht er aber im „Schatten“ des Bildwiederholspeichers (IRM) (siehe Bild 2). Durch das steuernde Programm muß nun gewährleistet werden, daß der IRM immer dann inaktiv wird, wenn der Bildschirminhalt nicht verändert werden soll. In diesem Moment kann der Prozessor auf den zweiten M022 zugreifen, um dort Daten oder Programme zu speichern.

Diese Arbeitsweise wird von den vom Hersteller gelieferten Sprachübersetzer BASIC und FORTH garantiert. Dem Anwender stehen damit 48 KByte RAM zur Verfügung.

Da in dem Beispiel der IRM und der zweite M022 den gleichen Adreßbereich belegen, wird durch die Modulprioritätskette verhindert, daß beide Speicherbereiche gleichzeitig aktiv sind /2/. In dieser Kette sind alle Speicher und Peripherieanschlüsse des Grundgerätes und alle Module hintereinander angeordnet. Ein aktiver Modul verhindert z. B., daß in der Kette danach angeordnete Module aktiv werden. Damit sind Kollisionen auf dem Prozessorbus ausgeschlossen.

In Tafel 2 sind die derzeit oder in Kürze verfügbaren Module und die Aufgabe der jeweiligen Modulsteuerung zusammengestellt. Der Modul M006 BASIC ist für den KC 85/2 entwickelt worden. Mit diesem Modul erreicht der KC 85/2 die Leistungsfähigkeit des KC 85/3. Zur Vollständigkeit wurde dieser Modul in Tafel 2 mit aufgeführt.

Der Beitrag wird mit Ausführungen zum Softwarekonzept des KC 85/3 in Mikroprozessortechnik 3/87 fortgesetzt.

Literatur

/1/ WP DD 223 272 A1 G06 F 13/04
Verfahren und Anordnung zur digitalen Informationsaufzeichnung und -übertragung

/2/ WP DD 220 433 A1 G06 F 12/06
Anordnung zur Vergrößerung des adressierbaren Speicherbereiches für Mikrorechner

Nutzerkatalog für Kleincomputer

Für die robotron-Kleincomputer KC 85/1 und KC 87 wird ein Katalog mit Anwendungslösungen vorbereitet. Hauptanliegen des Kataloges ist es, die nahezu unvermeidlichen Mehrfachentwicklungen von Anwendungslösungen durch gezielte und schnelle Information auf ein Minimum zu beschränken. Der Katalog dokumentiert sowohl Hardware- als auch Softwarelösungen, die von Anwendern beim Einsatz der Kleincomputer in den unterschiedlichsten Bereichen erarbeitet wurden und zur Nachnutzung angeboten werden. Er soll in geeigneten Abständen gedruckt werden und dann auf Anfrage ausleihbar bzw. gegen eine Schutzgebühr auch käuflich zu erwerben sein. Angebote von Lösungen zur Aufnahme in den Katalog bzw. Anfragen zu dokumentierten Lösungen sind zu richten an: VEB Robotron-Meßelektronik „Otto Schön“ Dresden, Abt. 1EKG, PSF 211, Lingnerallee 3, 8010

Dr. Gunter Kleinmichel

29. ZMMM: Messe der Computer

Aus dem großen Angebot an nachnutzbaren Lösungen zeigen wir eine Auswahl von Exponaten zur Computertechnik, die in vielen Bereichen Schwerpunktthema war.


Fotos: Weiß (10)

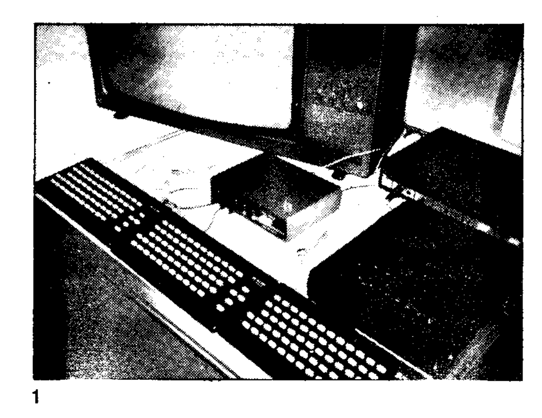
Für den Einsatz an Bildungs- und Betriebseinrichtungen (Computerclubs und -kabinette), die im Dialogbetrieb mit dem KC 85/2 arbeiten, ist die **Mehrtastaturbedienung** (Bild 1) gedacht. An einem KC 85/2 können über den Koppelbaustein (Bildmitte) bis zu 6 Tastaturen angeschlossen werden, die sich gegenseitig nicht beeinflussen. Die Methodik des Unterrichtes kann damit entscheidend verbessert werden.
Betriebsschule „Hermann Jahn" des VEB Mikroelektronik „Karl Marx" Erfurt, Binderslebener Landstr. 160, Erfurt, 5023

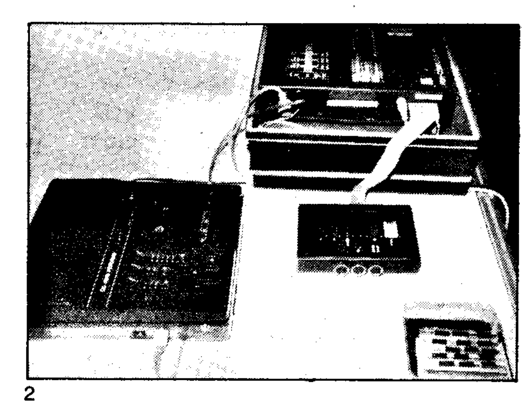
Das **Schaltkreisprüfgerät** (Bild 2) als Zusatzmodul zum Polycomputer ermöglicht es, integrierte Schaltungen mit niedrigem und mittlerem Integrationsgrad zu testen. Es können Bauelemente der DDR- und SU-Produktion auf ihre Funktionsfähigkeit vor dem Einbau objektiv geprüft werden. Die Prüfzeit läßt sich von 15 Minuten auf 5 Sekunden senken.
NVA, Postfach 14 413/6, Strausberg, 1260

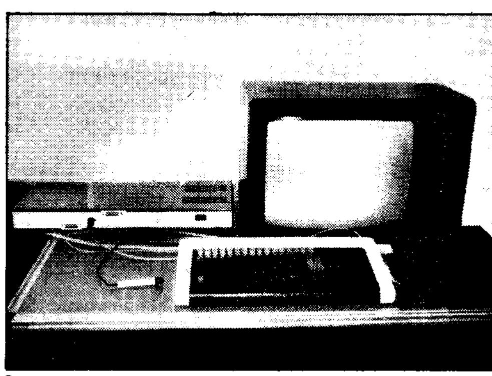
Computergrafik CG 86 (Bild 3) ist eine Leiterkarte im K-1520-Format zum Darstellen von Text und Grafik auf einem Farbmonitor. Sie besitzt Funktionsgruppen für Textspeicher (64 × 32 Zeichen), Farbspeicher (64 × 32 Farbfelder zu 8 × 8 Bildpunkten), Grafikspeicher (512 × 256 frei programmierbare Bildpunkte) sowie Anschlüsse für Lichtstift, MBG, Tastatur und weitere 4 CG 86.
Technische Hochschule Magdeburg, Sektion Technische Kybernetik und Elektrotechnik, Boleslav-Bierut-Platz 5, Magdeburg, 3010

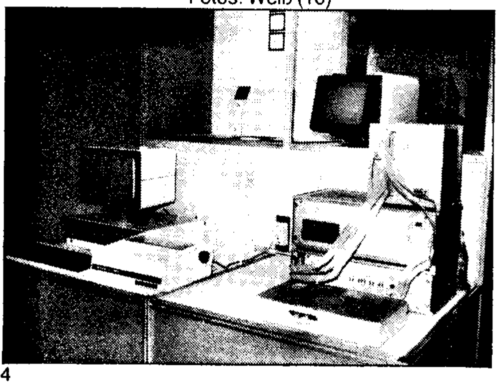
Für die Prozeßanalyse und Prozeßautomatisierung in der Zeitungsverpackungstechnik entstand die **CAM-Rechnerkopplung im Echtzeitbetrieb** (Bild 4). Durch den freizügigen Informationsaustausch zwischen Geräten der Zeitungsverpackungstechnik auf der Basis des an der TU Dresden entwickelten lokalen Rechnernetzes LOTUNET entstehen Vorteile bei der Produktionskontrolle und -steuerung; zum Beispiel durch Parallelverarbeitung von Auflagen und Realisierung von Adressierern.
Rationalisierungsmittelbetrieb der Zentrag, Abt. Automatisierungstechnik, Kohlgartenstr. 5/9, Leipzig, 7050

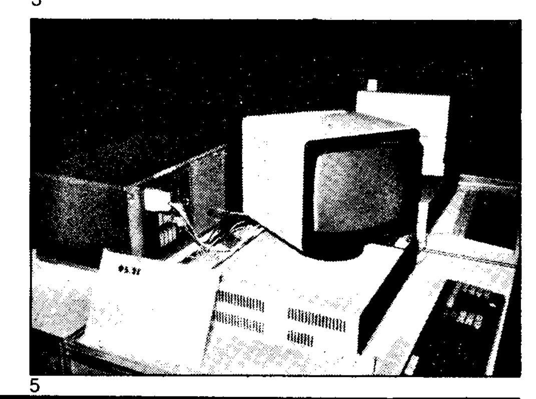
Als Prinziplösung zum Anschluß von Personal- und Bürocompu-

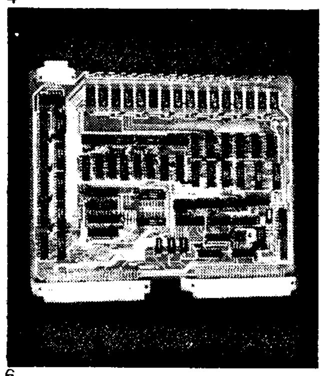

1


2

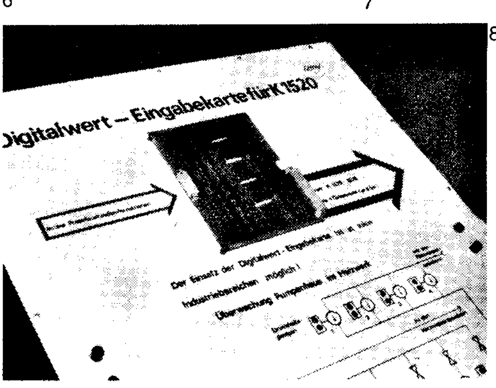

3


4

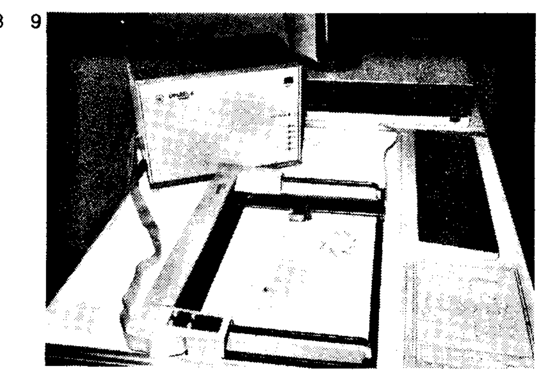

5

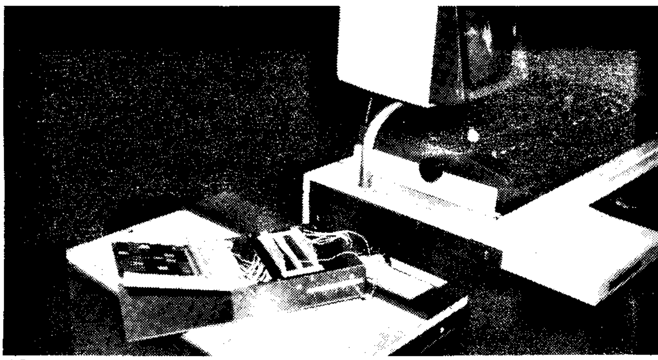

6


7



8


9


10

tern an das Datennetz der DDR wurde die **Terminalanschlußsteuerung** (Bild 5) entwickelt. Als Dienste werden für die Übertragung kurzer Mitteilungen die Operatorkommunikation und für Diskettendateien die Dateiübertragung bereitgestellt.
Ein Teilergebnis der Lösung ist der Aufbau eines BC/PC-Verbundes auf Basis der Terminal-Kommunikationseinrichtung ohne Nutzung des Datennetzes.
AdW der DDR, Institut für Informatik und Rechentechnik, Forschungsgruppe Netzzugriff, Rudower Chausse 5, Berlin, 1199

Eine auf der Basis von DDR-Bauelementen entwickelte **Farbgrafiksteckeinheit CGD** (Bild 6) für K-1520-OEM-Systeme bzw. Bürocomputer gestattet den Anschluß von handelsüblichen Farbfernsehgeräten (mittels Adapter, ebenfalls zur Nachnutzung angeboten) sowie anderer Farbmonitore. Die Auflösung beträgt 156 × 256 Bildpunkte in 8 Farben und 2 Helligkeitsstufen. Darüber hinaus steht unter UDOS CAD-Software als makrofähiger Grafik-Editor für Leiterplatten-, Schaltbild- und andere Grafikentwürfe zur Verfügung.
VEB Datenverarbeitungszentrum Karl-Marx-Stadt, BfN, PSF 729, Karl-Marx-Stadt, 9010

TESY (Bild 7) ist ein technologisches System miteinander gekoppelter Kleincomputer KC 85 /2 bzw. /3 nach dem Master-Slave-Prinzip. Die zwei Typen der dafür geschaffenen Steuermoduln (Slave-Modul und Master-Modul) werden über den Modulschaft des KC an den Bus angeschlossen und erfordern keine Hardwareveränderungen am Rechner. Mit den gegenwärtigen Moduln können bis zu 16 Schülerrechner (Slaves) mit dem Lehrerrechner (Master) zwecks Datentransfer gekoppelt werden.
NVA, Postfach 36017/B, Bad Düben, 7282

Die zum K-1520-Bus kompatible **Digitalwert-Eingabekarte** (Bild 8) erlaubt die Erfassung von 32 binären Prozeß-Zustandsinformationen. Die Eingabe der Digitalwerte erfolgt über 24-Volt-Stromschleifen, die mittels Optokoppler von der Rechner-Stromversorgung galvanisch getrennt sind. Der Anschluß der Peripherie erfolgt über eine 39polige Buchse; Abmessungen der Karte 215 × 170 mm^2.
Zählerreparaturwerk Oranienburg, Heidelberger Str. 32, Oranienburg, 1400

Uniplot UPL 861 (Bild 9) ist ein Plotrahmen, der auf die Zeichenfläche aufgesetzt wird. Daraus ergibt sich die Besonderheit, daß Zeichnungen angefertigt werden können, die größer als das Format des Plotrahmens (A3) sind, indem er auf der Zeichenfläche umgesetzt wird. Auflösung 0,025 m, max. Achsgeschwindigkeit 12 cms^{-1}, Schriftdarstellung max. 2 Zeichen/s bei Standardgröße, Schnittstelle z. Z. IFSS.
AdW der DDR, Zentrum für wissenschaftlichen Gerätebau, Rudower Chaussee 6, Berlin, 1199

Mit dem **Steckeinheitprüfgerät STEP 5100** (Bild 10) lassen sich Logik-Baugruppen der ESER- und Mikrorechentechnik mit maximal 240 Kontakten in Kopplung mit einem BC A 5130 als Steuerrechner quasistatisch prüfen. Die z. Z. 80 typenspezifischen Prüfprogramme sind auf Disketten gespeichert; Fehler werden auf dem Bildschirm angezeigt.
VEB Datenverarbeitungszentrum Dresden, Dr. Otto-Nuschke-Str. 20, Dresden, 8012

Hans Weiß

Fachtagung

Computer- und Mikroprozessortechnik '86

Am 28. und 29. Oktober 1986 fand diese Veranstaltung in Berlin mit etwa 500 Teilnehmern, u. a. aus der VR Bulgarien, der VR Polen, der Sowjetunion und der ČSSR statt. Veranstalter der jährlich durchgeführten Tagung war die wissenschaftliche Sektion Computer- und Mikroprozessortechnik im Fachverband Elektrotechnik der Kammer der Technik. Die wissenschaftliche Leitung oblag Mitarbeitern der Sektion Elektronik der Humboldt-Universität zu Berlin.
Inhaltlich befaßte sich die Fachtagung mit aktuellen Problemen der Computer-Wissenschaft, und insbesondere wurden Auffassungen zu Entwicklungstendenzen der Hard- und Softwareforschung sowie Übersichten über wichtige Applikationen vorgestellt. Eine Fachbuchausstellung des VEB Verlag Technik Berlin und des VEB Fachbuchverlag Leipzig sowie die Ausstellung neuer Erzeugnisse des VEB Kombinat Robotron mit dem 16-Bit-Mikrorechner A 7100 im Mittelpunkt bildeten einen niveauvollen Rahmen.
Neben den Plenarvorträgen wurden in folgenden Sektionen Übersichtsvorträge geboten:

A – Mikrorechnersysteme und Rechnernetze
B – Prozeßrechentechnik
C – CAD-Systeme
D – Programmiersprachen und Betriebssysteme
E – Schaltkreisentwurf.

Im Eröffnungsvortrag gab *Dr. Walter,* Direktor für Forschung und Entwicklung im VEB Kombinat Robotron, einen Überblick über das gegenwärtige und zukünftige Erzeugnisprofil.
Prof. Dr. Adler vom Zentrum für Informatik an der TU Dresden beschäftigte sich mit Entwicklungsrichtungen in der Software-Technologie. Dabei orientierte er nachhaltig auf die Verwendung moderner Sprachkonzepte, mit deren Hilfe transparente, portable, strukturierte und modularisierte Programme erarbeitet werden können. Insbesondere wurde auf Programmiersprachen wie MODULA 2 und PASCAL in einer effektiven und ausgewogenen Programmierumgebung verwiesen.
Zu Problemen moderner Rechnerarchitekturen erörterte *Prof. Dr. Jugel* vom VEB Robotron Meßelektronik Dresden Zellulare Systeme als alternatives Prinzip. Es gelang ihm mit seinem Vortrag, die Hörer für derartige Parallel-Architekturen zu interessieren, die sich von Modellen zur Struktur und Funktion des menschlichen Gehirns ableiten und deren technische Realisierung zum gegenwärtigen Zeitpunkt noch nicht möglich ist.
In weiteren Plenarvorträgen wurden von *Prof. Dr. Fristácki,* TU Bratislava, Übersichten zur Entwicklung der Mikroprozessor-Architekturen und von *Dr. Mackrodt,* VEB Kombinat Robotron, Zusammenstellungen zu Hardware-Unterstützungen von Programmiersprachen in Systemem der Künstlichen Intelligenz, überwiegend LISP-Prozessoren, gegeben.
Von den Übersichtsvorträgen kann, da der Verfasser eine Auswahl bezüglich der Teilnahme treffen mußte, nur punktuell genauer berichtet werden.
In der Sektion A unterzog *Dr. Kleinmichel* vom VEB Robotron Meßelektronik Dresden den gegenwärtig verfügbaren Kleincomputer aus seinem Betrieb einer schöpferischen Kritik und entwickelte daraus Anforderungen an ein Nachfolgesystem, die etwa einem heutigen Personalcomputer der unteren Leistungsklasse entsprechen.
Seitens der Sektion Elektronik der Humboldt-Universität zu Berlin wurde auf die Allgemeinheit und Flexibilität des Hardwarekonzepts des lokalen Rechnernetzes LANCELOT 1 hingewiesen. Darüber hinaus gab der Beitrag Aufschluß zur Struktur und Portabilität der erforderlichen Software, die für den Nutzer teils als Firmware und teils als Programm-Module, überwiegend unabhängig vom Betriebssystem, zur Verfügung gestellt werden kann.
Dr. Bell von der Sektion Mathematik der Humboldt-Universität zu Berlin untersetzte diese Ausführungen mit der Vorstellung eines Softwaresystems für ein lokales Rechnernetz mit LANCELOT-1-Hardware, in dem die höheren Softwareschichten mit Hilfe von MODULA 2 programmiert sind.
Sektion B war überwiegend mit Problemen der dezentralen Prozeßdatenverarbeitung befaßt. Daneben wurde eine Lösung für eine mehrkanalige, hochauflösende Digital-Analog-Umsetzer-Baugruppe für 16-Bit-Mikrorechnersysteme vorgestellt. In der Sektion C gelangten als Schwerpunkt Entwürfe für CAD-Systeme zur Darstellung, die auf verfügbarer Mikrorechentechnik

basieren und die teilweise auf relativ eng begrenzte Aufgabenstellungen zugeschnitten sind.
Ein besonderer Höhepunkt der Sektion D war zweifellos der Beitrag von *Prof. Dr. Janicki* vom Institut für mathematische Maschinen in Warszawa, über die Programmiersprache LOGLAN. Es zeigt sich, daß diese Sprache allen Anforderungen an ein modernes Software-Engineering entspricht und offenbar als Alternative zu PASCAL, MODULA 2 u. a. anzusehen ist. LOGLAN ist in der VR Polen weit verbreitet und beispielsweise auch auf dem IBM-PC implementiert.
Des weiteren informierten Mitarbeiter der TH Ilmenau über interessante BASIC-Spracherweiterungen in Richtung strukturierter Programmierung, Echtzeitverarbeitung und Behandlung paralleler Prozesse.
Guertzig, TH Ilmenau, warf in seinem Vortrag die Frage nach dem Einsatz der Programmiersprache PROLOG in der Steuerungstechnik auf. Er konnte dabei von erstaunlichen Teilerfolgen berichten, die auf einer völlig unkonventionellen Problembeschreibung im Sinne der Prädikatenlogik fußen.
Sektion E beschäftigte sich mit Fragestellungen des Schaltkreisentwurfs, wobei die Entwicklung von Siliconcompilern und die Hardware-Realisierung von FORTH-Systemen besondere Beachtung fanden.
Die nächste Veranstaltung wird voraussichtlich in Magdeburg stattfinden.

Prof. Dr. Zaremba

4. Fachtagung

Anwendung von Mikrorechnern in der Meß- und Automatisierungstechnik

Die Fachtagung „Anwendung von Mikrorechnern in der Meß- und Automatisierungstechnik" fand am 11. und 12. September 1986 im Rahmen der Wissenschaftlichen Tage 1986 der Technischen Hochschule „Otto von Guericke" Magdeburg (THM) statt. Sie wurde gemeinsam von den Wissenschaftsbereichen Prozeßmeßtechnik und Regelungstechnik/Prozeßsteuerungen der Sektion Technische Kybernetik und Elektrotechnik in Verbindung mit dem Fachausschuß Mikroprozessor-Interfacesysteme der KDT ausgerichtet. Mit über 500 Teilnehmern, darunter Gäste aus dem Ausland, und über 90 Vortragsmeldungen fand die Tagung eine sehr große Resonanz.
Die Veranstaltung diente in erster Linie der Vermittlung und dem Austausch von Erfahrungen, die bei der Mikrorechneranwendung zur Lösung von meß- und automatisierungstechnischen Aufgaben des Schwermaschinen- und Anlagenbaus (SAB), der Kraftwerkstechnik sowie bei der Steuerung fertigungstechnischer Prozesse und bei der Automatisierung chemischer und wasserwirtschaftlicher Prozesse mit Mikrorechnern gesammelt wurden. Zwei Plenar- und 9 Hauptvorträge gaben zunächst einen Überblick zum Stand und Trend der Meßwerterfassung mit Mikrorechnern sowie zu Mikrorechner-Automatisierungsanlagen.
Am zweiten Tag wurden in den Fachsektionen
1. Gerätetechnische Lösungen und Sensoren für die Prozeßmeßtechnik
2. Meß- und Auswerteverfahren für die Prozeßmeßtechnik
3. Algorithmen der automatischen Steuerung; Mikrorechner-Automatisierungsmittel
4. Anwendung von Mikrorechnern zur Aggregat- und Prozeßautomatisierung

insgesamt 47 Vorträge gehalten, zu denen es rege Diskussionen gab.
Nachfolgend wird lediglich über diejenigen Vorträge berichtet, die sich unmittelbar mit dem MR-Einsatz befaßten.
M. Seifart (THM) gab in seinem Plenarvortrag einen Überblick zum Stand und Trend der Meßwerterfassung mit Mikrorechnern. Ausgehend vom Vergleich verschiedener Grundstrukturen wurde als entscheidendste Auswirkung des Mikroprozessor- und Einchipmikrorechnereinsatzes der allmähliche Übergang zu dezentralen prozeßnahen Systemstrukturen mit digitaler serieller Informationsübertragung über einen einfachen Feldbus herausgestellt, deren Hauptbestandteile intelligente Module zur Meßgrößenerfassung und zur Beeinflussung des Prozesses sind. Beim Übergang zu intelligenten Systemkonzeptionen sind vor allem zwei Richtungen zu unterscheiden: 1. Meßwerterfassungsleiterkarten, die meist in MR oder MC einsteckbar sind (paralleler Bus) und 2. Intelligente prozeßnahe Prozeßkoppelmodule, die über einen seriellen Feldbus untereinander und mit einem Leitgerät verbunden sind. Zu beiden Varianten wurden an der THM realisierte Beispiele vorgestellt und der Einfluß des PC auf die Meßwerterfassung behandelt.



Foto: Locker

Die 6 Hauptvorträge erfaßten die gesamte Kette der mikrorechnergestützten Meßwerterfassung von der Sensorik über die Signalverarbeitung bis hin zu Sofwareproblemen. Den bereits im Plenarvortrag erläuterten Trend zu dezentralen prozeßnahen Systemstrukturen erläuterte *F. Güttler* (THM) am Beispiel eines digitalen Automatisierungssystems für die Kleinautomatisierung mit dezentralen Mikrorechnern (U 882).
W. Meiling (TUD) gab einen Überblick über serielle Bussysteme, wobei die gesamte Hierarchie vom internen Schaltkreisbus über Komponentenbusse, lokale Netzwerke bis zu den Prozeßbussen einbezogen wurde. *B. Michaelis* und *Ch. Wartini* berichteten über den Einsatz optoelektronischer Sensoren und Methoden der Signalauswertung unter besonder Berücksichtigung des Schwermaschinen- und Anlagenbaus. Zu Softwareproblemen bei intelligenten Meßwerterfassungssystemen referierte *D. Werner* (TUD). Er forderte mit Nachdruck, stärker auf höhere Programmiersprachen überzugehen und erforderlichenfalls zur Erfüllung kritischer Echtzeitforderungen kleinere Programmstücke nachträglich in einer maschinenorientierten Sprache zu implementieren.
Weitere Hauptvorträge befaßten sich mit dem Stand und Tendenzen der Sensorentwicklung (*P. Hauptmann,* THM) sowie mit Systembetrachtungen zu Meßwerterfassungsstrukturen (*H.-J. Dubrau,* IHD).
Von den in der stark besuchten Fachsektion 1 gehaltenen 14 Vorträgen behandelten 6 Vorträge den Einsatz des EMR U 882 zur intelligenten Meßwerterfassung.
Schwerpunkt der Sektion 2 waren Meß- und Auswertverfahren für die Prozeßmeßtechnik.
Der zweite Hauptkomplex der Tagung war automatisierungstechnischen Fragen des Mikrorechnereinsatzes gewidmet. *P. Neumann* (THM) sprach im Plenum über die Anforderungen des Innovationsprozesses bei Mikrorechner-Automatisierungsanlagen an die wissenschaftlichen Untersuchungsmethoden.
Die Hauptvorträge zur Thematik „Automatisierungstechnische Fragen des Mikrorechnereinsatzes" hatten das Ziel, über Erzeugnisse der Automatisierungsindustrie der DDR und über Beispiele in den sozialistischen Staaten zu informieren. Behandelt wurden u. a. das Prozeßleitsystem audatec, das prozeßnahe Steuerungs- und Regelungssystem EAW-Compact S 2000 und das MR-Automatisierungsanlagensystem MIK 2000 C aus dem Institut für Technische Kybernetik und Robotertechnik der Bulgarischen Akademie der Wissenschaften.
Die Mehrzahl der in den Fachsektionen 3 und 4 gehaltenen Vorträge befaßte sich mit dem Einsatz von Mikrorechnern in Prozeßleitsystemen, in der Steuerungstechnik, in der Energietechnik, im Verkehrswesen und in der Kerntechnik.
Die Vorträge und Posterbeiträge sind in zwei Tagungsbänden (etwa 200 S.) zusammengefaßt. Ausgewählte Beispiele werden darüber hinaus in der Wissenschaftlichen Zeitschrift der THM erscheinen. Die 5. Fachtagung findet voraussichtlich 1989 statt.

Prof. Dr. Seifart

Am 8. Februar 1957 verstarb der bedeutende ungarisch-amerikanische Mathematiker, Physiker und Ingenieur John von Neumann. Von Neumann begründete die moderne Funktionalanalysis und die Spieltheorie. Er verfaßte wichtige Arbeiten über die Gruppentheorie und förderte entscheidend die Entwicklung moderner Rechenautomaten. Aus Anlaß seines 30. Todestages haben wir die nachfolgende Broschüre rezensiert.

Die Rechenmaschine und das Gehirn

Von J. v. Neumann, R. Oldenbourg Verlag München 1986, 5. unveränderte Auflage, 80 S.

Als Entwickler des Versuchsrechners JONIAC (1945) und Mitentwickler der 1946 leistungsstärksten digitalen Rechenanlage ENIAC mit etwa 22 000 Röhren, wurde John von Neumann zum „Vater" der auch heute noch weltweit genutzten Princeton-Rechner-Architektur, deren wesentliche strukturelle Merkmale in der Existenz von den drei Funktionsblöcken Rechenwerk, Steuerwerk und Speicher bestehen. Ihre Verbindung wird durch einen für Befehle und Daten einheitlichen Bus realisiert. Außerdem ist wesentlichstes funktionelles Merkmal der streng sequentielle, adreßorientierte Steuerungsablauf der Programmabarbeitung (Steuerflußprinzip).
Merkmale des *Steuerflußprinzips* dieser J.-v.-Neumann-Architektur sind auch mit dem Namen Konrad Zuse verbunden, der bereits 1941 mit dem ersten programmgesteuerten Rechenautomaten der Welt, dem Z3, dieses Grundprinzip des maschinellen Rechnens erfunden hatte. Die kleine Schrift *Die Rechenmaschine und das Gehirn* ist John von Neumanns letzte wissenschaftliche Arbeit. Sie wurde geschrieben in der Gewißheit des bevorstehenden Todes und ist in gewissem Sinne als Erkenntnisvermächtnis für seine letzte – nicht mehr gehaltene – Ehrenvorlesung vor dem Silliman Lecture Commitee der Yale Universität (USA) im Jahre 1957 zu verstehen.
Das Büchlein J. v. Neumanns beschreibt zur Hälfte etwa die wesentlichsten Parameter „moderner" Rechenanlagen (1957) (Relais, Röhren, Dioden, Ferritkerne, Transistoren) mit der Prognose, in den folgenden 10 Jahren Operationszeiten von 10^{-8} bis 10^{-9} Sekunden erreichen zu können, dabei liegt „die Anzahl der aktiven Elemente größerer, modernerer Maschinen etwa zwischen 3000 und 30 000", wobei etwa zwischen 200 und 2000 aktive Elemente zu Speicherfunktionen dienen. Das eigentlich interessante in der Arbeit Neumanns aber ist der Vergleich des Computers mit dem Gehirn. Die Reaktionszeit eines Neurons liegt zwischen 10^{-4} und 10^{-2} Sekunden. Mit etwa 10^{10} Neuronen bei gut 10^3 cm^3 Gehirnvolumen beträgt der Raumbedarf des einzelnen Neurons 10^{-7} cm^3. Der Gesamtenergiebedarf des Gehirns ist etwa 10 W.

Aus diesen Angaben folgt aus der Sicht vor 30 Jahren:
– Der Raumbedarf der künstlichen Elemente ist 10^8- bis 10^9mal so groß als jener der natürlichen.
– Der Energieaufwand ist ebenfalls um den Faktor 10^8- bis 10^9mal so hoch.
– Die Bearbeitungszeit eines Signals dagegen beträgt nur den 10^{-n}fachen Teil.
Daraus folgt die entscheidende Aussage der Schrift: „Große und gut organisierte natürliche Automaten (Gehirn, M. R.) werden daher wahrscheinlich in großem Maße parallel arbeiten, während große und gut organisierte künstliche Automaten (Computer, M. R.) mehr zum Serienbetrieb tendieren werden".
Die Schrift John von Neumanns, dem Pionier der Rechentechnik, ist anregend und gut zu lesen.
Prof. Dr. M. Roth

UNIX-Buch in der UdSSR erschienen

Vor kurzem erschien im Moskauer Verlag Radio i Swjas die russische Übersetzung des Buches „Einführung in das Betriebssystem UNIX" von M. Banahan und A. Rutter. Das Vorwort von I. G. Schostakow und W. W. Leonas geben wir an dieser Stelle leicht gekürzt wieder.

Neuerdings wächst das Interesse am Betriebsystem UNIX und an kompatiblen Betriebssystemen beträchtlich. Das liegt an der vollen Kompatibilität zur Programmiersprache C und an der großen Anwendungsbreite von UNIX, das inzwischen international zum Standard-Betriebssystem für 16-Bit-Rechner geworden ist.
UNIX-Versionen laufen auf einer großen Anzahl verschiedener EDVA – von den 8-Bit-Rechnern auf der Basis der Intel-8080-Mikroprozessoren bis zum Superrechner Cray-1. Darüber hinaus wird das Betriebssystem UNIX über verhältnismäßig einfache Interfaces realisiert, und es garantiert dem Nutzer zahlreiche Programmiermittel. Damit wird es den modernen Anforderungen an Computersoftware gerecht.
Auch in der UdSSR besteht ein großer Bedarf an UNIX und an kompatiblen Varianten. 1983 legte das Institut für Elektronische Rechner die vorbereitende Version INMOS vor (Instrumentelles **M**obiles **O**perationssystem), die auf Kleinrechnern CM-4, CM-1420, CM-1644 und Elektronika 100/25 lauffähig ist. Gegenwärtig arbeitet das Institut an einer zur Version 7 von UNIX (Standard-UNIX) kompatiblen INMOS-Variante. Die zunehmende Anwendung von UNIX führt auch in der UdSSR zur Herausgabe zahlreicher Literatur über das Betriebssystem (BS) und seine Programmiermöglichkeiten. Eines der ersten erschienenen Bücher zum Thema ist die vorliegende Publikation.
Die Originalausgabe wurde 1982 gerade für jene Anwender herausgebracht, die keine Programmiererfahrungen mit Computern haben und aus bestimmten Gründen UNIX studieren und anwenden wollen. Die Autoren stellten sich – mit Erfolg – der schweren Aufgabe, Lehrbuch und Anwendungshilfe zugleich vorzulegen. Grundlage für diese Doppelfunktion ist die Programmierübersicht zum UNIX als Anleitungsmaterial (UNIX Programmers Manual), ohne die der unerfahrene Leser nicht auskommen wird. Auf vergleichsweise geringem Raum bieten die Autoren dem so vorbereiteten Nutzer wahrhaft eine Fülle von Informationen und Programmiererfahrung an. Alle 10 Kapitel des Buches durchziehen die Organisationsprinzipien und die Funktionsgrundlagen des BS. Um diese Einheitlichkeit zu erreichen, arbeiten die Autoren stark mit Anwendungsbeispielen, die für die heute vorliegenden Versionen des BS nutzbar sind. Gesondert erklärt sind die Kommandos, Editor, Shell-Syntax, Systemaufrufe, Instrumentarien und Standard-Bibliotheksfunktionen. Die Autoren verfahren konsequent nach dem Leitspruch „Vom einzelnen zum allgemeinen – erst die Praxis, dann die Theorie". Doch auch der wenig geübte Anwender wird manchmal Schwierigkeiten haben, wenn Grundbegriffe vorausgesetzt werden müssen. Insgesamt ist das Buch aber leichtverständlich, anschaulich, teilweise sogar salopp geschrieben. Das sichert nicht immer die nötige Darstellungsstrenge, dient aber in seiner Faßlichkeit dem angesprochenen Leserkreis mehr als ein sachlich-trockener Stil. Und es schmälert keinesfalls den guten Gesamteindruck des Buches, das auch in russischer Sprache zahlreiche Leser finden wird, nicht zuletzt, weil es nützlich und aktuell ist.

Mega-Bit

Unter diesem Titel brachte der Verlag Junge Welt ein von der Redaktion Jugend + Technik erarbeitetes Journal zur ZMMM '86 heraus. Themen waren unter anderem Siliziumkristallzüchtung in Freiberg – SMD-Technik; Oberflächenmontage für Chips – Flexibles Fertigungssystem für Robotermotoren – Personalcomputer PC 1715 aus Sömmerda – FDJ-Computerklub Schwerin vorgestellt – BASIC-Kurzlehrgang – JU + TE-Lexikon Schlüsseltechnologien – Computergrafik für Forschung und Konstruktion.

Foto: Gratschow



BASIC für Mikrorechner

Programmentwicklung, Sprachelemente, Anwendungen von D. Werner, VEB Verlag Technik, Berlin, 1986

Im Vorwort bemerkt der Autor, daß die Sprache in wenigen Tagen oder höchstens einer Woche erlernbar ist. Man könnte dann meinen, daß eine Darstellung auf 240 Seiten nicht gerechtfertigt sei. So ist es aber nicht. Am systematischen Vortrag des Stoffes ist zu erkennen, daß der Autor über pädagogische Erfahrungen verfügt, daß ihm die Schwierigkeiten bekannt sind,
(Fortsetzung auf S. 64)

Fengler, W.; Roth, M.
Einchip-Rechner-Schaltkreise
Mikroprozessortechnik, Berlin, 1 (1987) 2, S. 37
Einchip-Rechner bilden minimale Mikrorechner-Konfigurationen mit allen wesentlichen Funktionen in einem Schaltkreis. Die Anwendung erfolgt als Geräterechner, als Mehrrechnerschaltkreis in modularen Systemen mit serieller oder paralleler Kopplung und als Makro-CPU.
Der Beitrag erläutert Architekturmerkmale, typische Anwendungen und Entwicklungsmittel von Einchip-Mikrorechnern (EMR).

Fengler, W.; Roth, M.
Схемы однокристальных ЭВМ
Mikroprozessortechnik, Berlin, 1 (1987) 2, стр. 37
Однокристальные ЭВМ образуют минимальные конфигурации микро-ЭВМ со всеми важными функциями в схеме. Применение происходит как машинная ЭВМ, как многомашинная схема в модульных системах с последовательной или параллельной связью и как макро-центральное устройство обработки данных.
Статья поясняет свойства архитектуры, типичные случая применения и средства проектирования однокристальных микро-ЭВМ.

Fengler, W.; Roth, M.:
Single-Chip Computers
Mikroprozessortechnik, Berlin, 1 (1987) 2, pp. 37
Single-chip computers represent minimal microcomputer configurations containing all the essential functions within one integrated circuit. They are applied as OEM computers, multicomputer integrated circuits in modular systems with serial and parallel coupling, and as macro CPU. The authors explain architectural features, typical applications and developing means of single-chip microcomputers.

Michaelis, B.; Maaß, R.
Meßwerterfassung mit CCD-Sensoren
Mikroprozessortechnik, Berlin, 1 (1987) 2, S. 41
Ausgehend vom allgemeinen Problem der optoelektronischen Informationsgewinnung mit Hilfe von CCD-Sensoren wird die Erfassung ausgewählter Meßgrößen durch eine geeignete Auswerteelektronik erläutert. Bei diesem Konzept verkürzt sich die Zeit zur Meßwertbereitstellung erheblich gegenüber der unmittelbaren rechentechnischen Auswertung des gesamten gewonnenen Bildes.

Michaelis, B.; Maaß, R.
Сбор измеряемого значения с датчиками на основе ПЗС
Mikroprozessortechnik, Berlin 1 (1987) 2, стр. 41
Исходя из общей проблемы оптоэлектронного получения информации с датчиками на основе ПЗС пояснен сбор выбранных измерительных величин при помощи пригодной электроники оценки. При этой работе время выборки измеряемых значений значительно сокращается по сравнению с непосредственной вычислительнотехнической оценкой всего получаемого изображения.

Michaelis, B.; Maaß, R.:
Measurement Value Acquisition by Means of CCD Sensors
Mikroprozessortechnik, Berlin 1 (1987) 2, pp. 41
Starting from the general problem of opto-electronic information acquisition by means of CCD sensors the author explains the acquisition of selected measurement quantities by a suited electronic evaluation circuit. Due to this conception the time for providing the measurement value reduces considerably compared to the direct computer evaluation of the whole image obtained.

Dorfmüller, L.; Despang, H. G.
Entwicklungsunterstützung für ein 16-Bit-Mehrmikrorechnersystem
Mikroprozessortechnik, Berlin, 1 (1987) 2, S. 51
Zum Test von größeren Programmen auf bereits funktionsfähiger Hardware dienen Monitore. Mittels einer Kopplung ist die Peripherie eines Cross-Systems nutzbar, und die Programmentwicklung kann effektiver gestaltet werden. Einen leistungsfähigen Monitor für I16-Systeme mit einer Kopplung zu einem 8-Bit-Entwicklungssystem und seine Nutzung an einem Zweitrechnersystem beschreibt der Beitrag.

Dorfmüller, L.; Despang, H. G.
Поддержка при развитии 16-тиразрядной многомикровычислительной системы
Mikroprozessortechnik, Berlin, 1 (1987) 2, стр. 51
Для проверки более крупных программ на основе уже работоспособного аппаратурного обеспечения служат мониторы. При помощи связи возможно использование периферии кросс-системы, и развитие программ может быть оформлено более эффективно. В статье описаны мощный монитор для 16-разрядных систем со связью в 8-разрядную систему проектирования и его использование на второй вычислительной системе.

Dorfmüller, L.; Despang, H. G.:
Development Support for a 16-Bits Multi-Microcomputer System
Mikroprozessortechnik, Berlin, 1 (1987) 2, pp. 51
In testing large program systems on a hardware ready for use monitors are applied. By means of a coupling the periphery of a cross system can be utilized, thus the program development becoming more effecient. The authors describe a high performance monitor for I16 systems with a coupling to an 8-bits development system as well as the use of this monitor in connection with a second computer system.

Götze, B.; Meusel, K.-H.
Personalcomputer in der Meßtechnik
Mikroprozessortechnik, Berlin, 1 (1987) 2, S. 54
Moderne leistungsfähige Meßgeräte, Meß- und Prüfsysteme kommen ohne integrierte Rechnerleistung nicht mehr aus. Im Beitrag werden dazu Lösungsmöglichkeiten erörtert. Qualitativ neue Lösungen für die Realisierung derartig „intelligenter" Meßtechnik ergeben sich bei der Nutzung von Personalcomputern, die inzwischen in allen Bereichen der Volkswirtschaft in immer größerer Breite zur Verfügung stehen. Im Beitrag wird insbesondere von den Anforderungen der modernen Logikanalyse an diese Fragen herangegangen.

Götze, B.; Meusel, K.-H.
Персональные компютеры в измерительной технике
Mikroprozessortechnik, Berlin, 1 (1987) 2, стр. 54
Современные мощные измерительные приборы, измерительные и испытательные системы не могут работь больше без интегрированной производительности ЭВМ. В данной статье обсуждены возможности решения. Качественно новые решения для реализации такой „интеллектуальной" измерительной техники получаются при использовании персональных компютеров, имеющихся в промежуточное время во всех областях народного хозяйства все время в более полной ширине. В данной статье этим вопросам приближают в частности исходя из требований современного логического анализа.

Götze, B.; Meusel, K.-H.:
Personal computers in Measurement Engineering
Mikroprozessortechnik, Berlin, 1 (1987) 2, pp. 54
Modern efficient measuring devices, measurement and test systems don't manage without integrated computer support. In this contribution the authors deal with corresponding technical possibilities. Solutions which are qualitatively new result from using personal computers largely available now in all fields of economics. The authors approach these questions particularly from the view point of the requirements of modern logic analysis.

(Fortsetzung von S. 63)

die derjenige hat, der sich zum erstenmal mit der Programmierung eines Rechners und mit Algorithmendarstellungen beschäftigt. Das Buch ist ein Lehrtext für die Hand des Anfängers! Für gewisse Abschnitte werden Vorkenntnisse benötigt. Diese Abschnitte kann der Anfänger überlesen. Es wird zuerst eine Einführung in allgemeine Fragen der Programmentwicklung gegeben, dann die Arbeitsschritte zum Anlegen eines Quelltextes und schließlich die Organisation des Übersetzens und Abarbeitens. Die Programmentwicklung mittels BASIC schließt sich an: Editieren, Abarbeiten, Debugging, Modifizieren, Optimieren.
Besonders ausführlich diskutiert werden die Darstellung des Datentransfers und die Nutzung externer Zusatzspeicher – erfahrungsgemäß haben Lernende damit Schwierigkeiten.
Eine Sammlung von Programmen zu unterschiedlichen Themen rundet den Lehrtext ab.

Dr. G. Paulin

BASIC

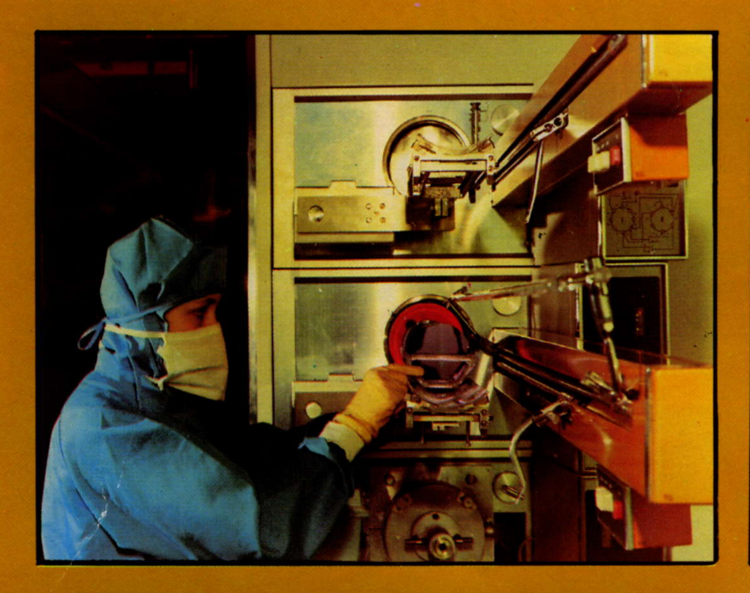
Einführen in das Programmieren
von R. Hopfer und R. Müller
VEB Fachbuchverlag Leipzig
1986

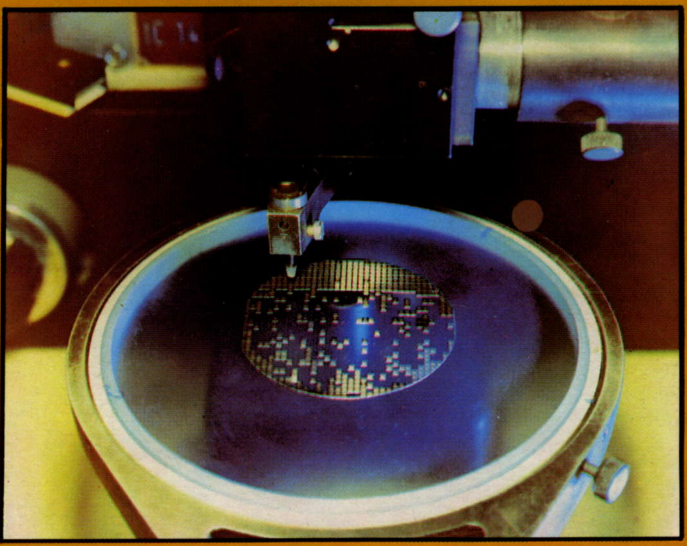
Im Vorwort haben die Verfasser formuliert: „Das Hauptanliegen der Broschüre besteht darin, eine geschlossene Darstellung der Lösung von einfachen Problemstellungen der Informationsverarbeitung mit Hilfe der Programmiersprache BASIC zu geben ... Die Darstellung des Stoffes ist so gewählt, daß auch Programmieranfänger sich im Selbststudium die Programmiersprache BASIC aneignen können."
Der thematische Aufbau der Broschüre unterstützt diese Zielstellung. Nach einer kurzen Einführung, die eine Übersicht über Aufbau und Programmierung eines Computers gibt, wird die eigentliche Arbeit mit dem BASIC-Interpreter erläutert. Erklärt werden BASIC-Sprachelemente, Anweisungen und Standardfunktionen. Eigene Kapitel gibt es auch für die Arbeit mit Feldern, Wertzuweisung mit Datenliste, Unterprogrammtechnik und vom Anwender definierbare Funktionen. Zahlreiche Beispiele erleichtern das Kennen- und Beherrschenlernen der Sprache BASIC.
Etwas zu kurz gekommen scheint der Abschnitt über Programmtest und Fehleranalyse, denn erfahrungsgemäß haben Anfänger bei Fehlersuche und -diagnostik die größten Schwierigkeiten.

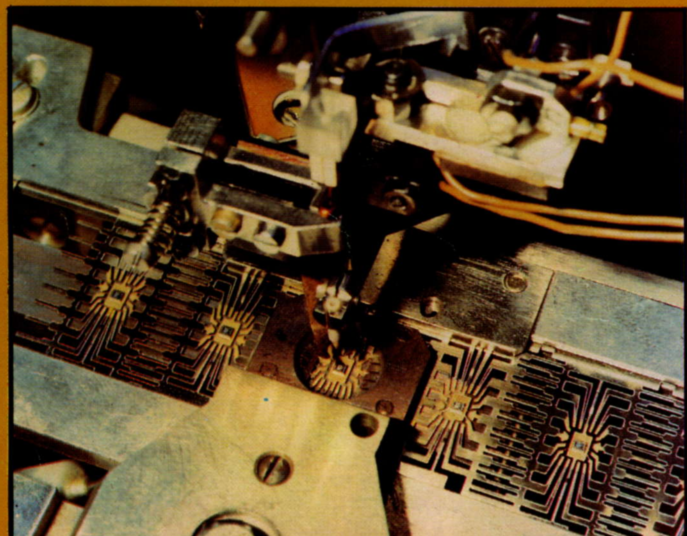
I. Paszkowsky

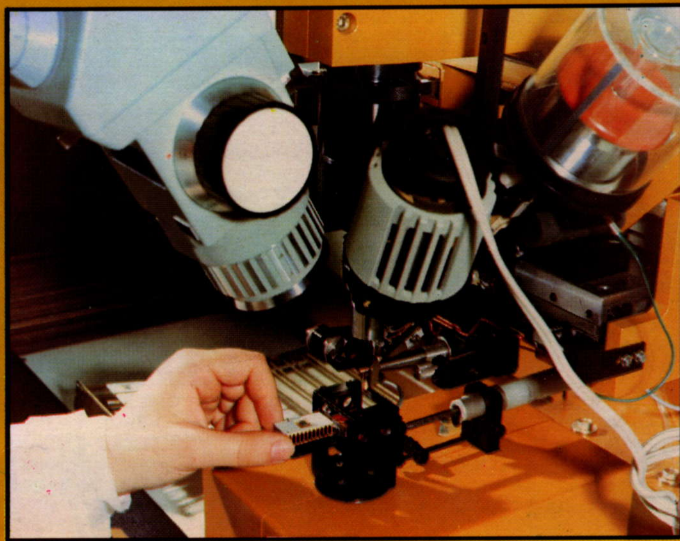
ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 2
S. 33–64

49 837 2 109 123 498
NV A
1260 9005 0121 PZV 20/43810





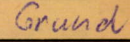




Schlüsseltechnologie Mikroelektronik

Die DDR gehört zu den wenigen Ländern der Welt, die derzeit in der Lage sind, mikroelektronische Bauelemente zu entwickeln, zu produzieren, ausgewählte Vormaterialien herzustellen und hochwertige technologische Ausrüstungen dafür zu fertigen.
Gegenwärtig sind mehr als hunderttausend hochqualifizierte Werktätige mit solchen Aufgaben befaßt.
Dabei werden die Kombinate Carl Zeiss JENA und Mikroelektronik Erfurt zu Zentren der Hochtechnologien ausgebaut. Auf der Grundlage neuer Basistechnologien wird das vorhandene Schaltkreissortiment erweitert. Die Entwicklung und Fertigung erfolgt in modernen, nach neuesten technologischen Gesichtspunkten eingerichteten Forschungslabors und Produktionstätten.

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK



MP

Heft 3 · 1987

Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0233-2892

P 8000 – universelles Programmier- und Entwicklungssystem

Lokale Rechnernetze mit OSI-Architektur

KC 85 als interaktives grafisches Display für den PC 1715

Die Tafel zeigt eine Auswahl von WEGA-Standardprogrammen des P8000-Mikrorechnersystems.

Lesen Sie dazu bitte den Beitrag ab Seite 68.

Anmerkung: Die mit einem S gekennzeichneten Systemprogramme sind sogenannte Superuser-Kommandoprogramme, über die nur der WEGA-Systemprogrammierer verfügen kann.

SYSTEMZUGRIFFSKONTROLLE

login		*UNIX-Nutzerzuschaltung*
newgrp		*Ändern der Gruppenzugehörigkeit*
passwd		*Schlüsselwortvergabe*
su		*Erweitern Benutzerzugriffsrechte*
sync	*S*	*Aktualisieren Superblock*

SYSTEMVERWALTUNG/STATUSINFORMATION

date		*Ausgabe Datum und Uhrzeit*
dd		*Konvertieren/Umkopieren*
du		*Ermitteln Plattenspeicherbelegung*
icheck	*S*	*Dateisystemkonsistenzüberprüfung*
learn		*UNIX-Einführungslernprogramm*
man		*Ausgabe Kommandobeschreibung*
mkfs	*S*	*Anlegen UNIX-Dateisystem*
mknod	*S*	*Anlegen Ein-/Ausgabedatei*
mount	*S*	*Mount/Dismount Dateisystem*
ps		*Prozeßstatusermittlung*
pstat	*S*	*Ausgabe Systemtabellen*
quot	*S*	*Ausgabe Dateisystemnutzerblöcke*

TERMINAL- UND DRUCKERARBEIT

echo		*Echoausgabe*
lpr		*Druckerausgabeprogramm (spooler)*
stty		*Setzen Terminalparameter*
tabs		*Setzen Terminaltabulatoreinstellung*
tty		*Ermittlung Terminalnamen*
who		*Wer arbeitet am System?*

KOMMANDOINTERPRETER

csh		*C-Shell-Kommandointerpreter*
sh		*Shell-Kommandointerpreter*

DATEIVERWALTUNG

cat		*Dateiausgabe*
cd		*Wechsel Arbeitsdirectory*
chmod		*Setzen Dateizugriffserlaubnisbits*
chown		*Wechsel Besitzerzugehörigkeit*
chgrp		*oder Gruppenzugehörigkeit*
clri	*S*	*Löschen i-node*
cmp		*Vergleich zweier Dateien*
comm		*Suchen gleicher Zeilen in Dateien*
cp		*Kopieren von Dateien*
crypt		*Verschlüsseln/Entschlüsseln*
dcheck	*S*	*Konsistenzprüfung Directory*
df	*S*	*Ausgabe freier Dateisystemblöcke*
diff		*Ermittlung von Dateiunterschieden*
file		*Ermittlung Dateityp*
find		*Suchen von Dateien*
ln		*Herstellen Dateilinkverbindung*
ls		*Ausgabe Directoryinhalt*
mkdir		*Anlegen Directory*
mv		*Verschieben/Umbenennen von Dateien*
od		*Dumpausgabe*
pwd		*Ausgabe Arbeitsdirectory*
rm		*Löschen von Dateien und*
rmdir		*Directories*
split		*Aufsplitten Datei*
sum		*Bestimmung Blockanzahl/Prüfsumme*
tail		*Übergehen von Dateiteilen*
touch		*Aktualisierung Dateidatum/-zeit*

KOMMUNIKATION

LOAD		*Programm vom P8000 in den Emulator laden*
SEND		*Programm vom Emulator in das P8000 übertragen*
getfile		*Einlesen von Dateien von einem Remote-System*
local		*Rückkehr zum lokalen Remote-System*
putfile		*Übertragung von Dateien zu einem Remote-System*
remote		*Zuschaltung übergeordnetes WEGA/UNIX-System*
cu		*UNIX-Rechnerkopplung*
enroll		*Schlüsselwortvergabe,*
xsend		*Senden und Empfangen*
xget		*geheimer Post*
mail		*Senden und Empfangen von Post*
mesg		*Nachrichtenempfangssperre*
uucp		*UNIX-UNIX Dateikopierprogramm*
uux		*UNIX-UNIX Kommandoausführung*
wall	*S*	*Schreiben an alle UNIX-Nutzer*
write		*Schreiben an andere Nutzer*

SOFTWAREPROJEKTUNTERSTÜTZUNG

ar		*Archiv- und Bibliotheksverwaltung*
cal		*Kalenderausgabe*
calendar		*automatisierter Terminkalender*
make		*Programmerzeugungshilfe*
tar		*Magnetbandarchivar*

PROGRAMMABARBEITUNGSUNTERSTÜTZUNG

at		*zeitgesteuerte Kommandoabarbeitung*
expr		*Berechnung arithmetischer Ausdrücke*
kill		*Prozeßabarbeitungsabbruch*
nice		*Wechsel der Abarbeitungspriorität*
nohup		
prof		*Ausgabe Programmabarbeitungsprofil*
sleep		*Abarbeitungspause*
tee		*Ein-/Ausgabepipevermittlung*
time		*Bestimmung Ausführungszeit*
true		*erzwungener Exitstatus TRUE/FALSE*
units		*Einheitenumrechnungsprogramm*
wait		*Warten auf Hintergrundprozeß*

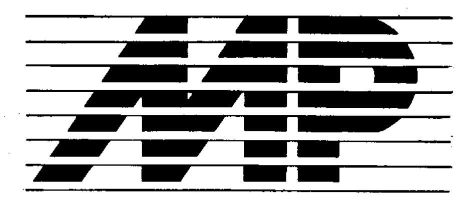
PROGRAMMIERSPRACHEN

adb		*interaktiver Programmtestdebugger*
as		*Assemblerprogrammübersetzer (PLZ/ASM)*
bc		*Arithmetik-Tischrechnerprogramm*
cas		*WEGA-Standardassembler (segmentiert/nichtsegmentiert)*
cb		*C-Programm Formatierungshilfe*
cc		*C-Compiler (nichtsegmentiert)*
dc		*Tischrechnerprogramm*
ld		*Lader (nichtsegmentiert)*
lex		*Programmgenerator für lexik. Analyse*
lint		*C-Syntaxprüfer*
lorder		*Ermittlung von Objektprogrammbezügen*
m4		*Makroprozessor*
nm		*Symboltabellenausgabe*
plzsys		*PLZ/SYS-Compiler*
plzcg		*PLZ/SYS-Kodegenerator*
scc		*C-Compiler (segmentiert)*
sld		*Lader (segmentiert)*
size		*Ausgabe Objektprogrammgröße*
strip		*Entfernen Symboltabelle*
yacc		*Compiler-Compiler*

TEXTVERARBEITUNG

awk		*Textfilterprogramm*
col		*Spaltendrucktextfilter*
deroff		*Entfernen aller Formatierungsbefehle*
ed		*Texteditor*
eqn		*Ausgabe mathematischer Formeln*
ex		*Texteditor*
grep		*Durchmustern von Dateien*
egrep		*erweitertes Durchmustern*
fgrep		*beschleunigtes Durchmustern*
join		*identifikationszeichenbedingte Ausgabe*
look		*Suchen von Dateizeilen*
pr		*formatierte Dateiausgabe*
ptx		*Wortindexerstellung*
sed		*Datenstromeditor*
sort		*Sortierprogramm*
spell		*Suchen Rechtschreibfehler*
tbl		*Tabellenformatierungsprogramm*
tr		*Zeichenkettenkonvertierung*
troff		*Textformatierung*
nroff		
uniq		*Löschen doppelter Dateizeilen*
vi		*bildschirmorientierter Texteditor*
wc		*Wortzählungsprogramm*

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0233-2892



Mikroprozessortechnik, Heft 3 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR-1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 0112228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)

Titelfoto DEWAG – Weber

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

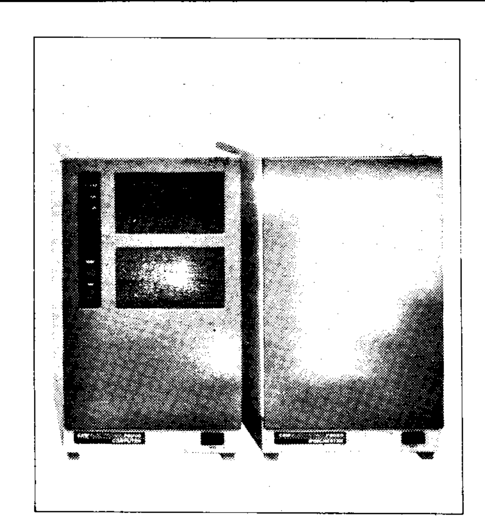
Redaktionsschluß: 26. Januar 1987

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

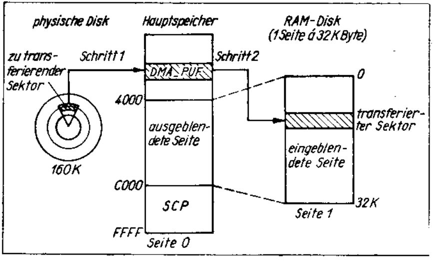
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tisku Praha, Slezská 11, 12000 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 88547 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavacko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hanoi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR-7010, und Leipzig Book Service, Talstraße 29, DDR-7010 Leipzig



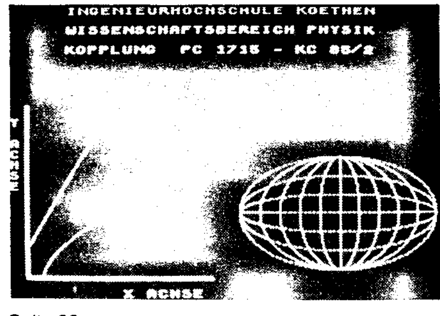
Seite 68

offenes System — offenes System
Schicht — Protokolle
7 Anwendungsschicht
6 Darstellungsschicht
5 Sitzungsschicht
4 Transportschicht
3 Netzschicht
2 Verbindungsschicht
1 Physikalische Schicht
Physikalisches Übertragungsmedium

Seite 74



Seite 83

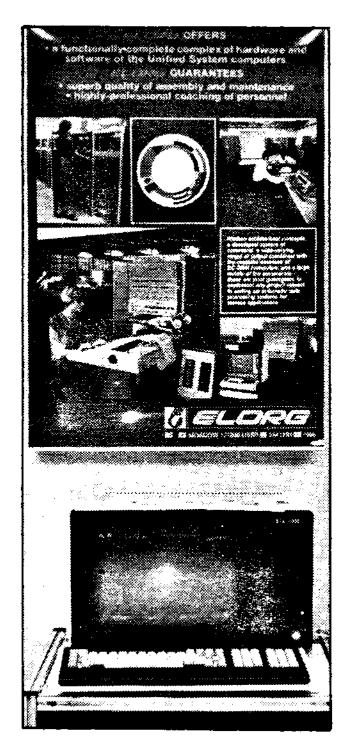


Seite 88

Inhalt

ELORG-Ausstellung in Berlin

Im Dezember vergangenen Jahres stellte das sowjetische Außenhandelsunternehmen V/O Elektronorgtechnika (ELORG) in Berlin Erzeugnisse der sowjetischen Elektronikindustrie vor. So waren u. a. elektronische Bauelemente, Taschenrechner und das intelligente Terminal WTA 2000-1 (Bild 1) zu sehen. Das alphanumerische Terminal kann im Offline- oder Online-Betrieb arbeiten. Das Gerät ist mit unterschiedlichen Schnittstellen und auch mit Grafikausstattung lieferbar.
Als Neuheiten präsentierte ELORG die Gate-Array-Schaltkreise K 1810 WM 1 und K 1520 XM 1 und die Speicherschaltkreise K 565 RU 7 mit 256 K RAM und K 132 RU 10 mit 64 K RAM (Bild 2). Weiterhin informierten Experten über die technischen Parameter der ESER-Anlage EC 1066 und des Multiplexers EC 8378.
Basis des Exportprogramms von ELORG sind die Rechenanlagen der ESER- und der SKR-Reihe. Zur Zeit arbeiten mehr als 1500 Rechenanlagen des ESER und des SKR in 18 Ländern der Welt. Zum Lieferumfang gehört auch Anwendersoftware – gegenwärtig werden etwa 200 Programmpakete für ESER- und SKR-Anlagen angeboten. MP



1

Fotos: Paszkowsky (2)

LSI–Speicher NEU
RAM256K
K565PY7
K132PY10
RAM64K

2

Datenbank für wissenschaftlich-technische Informationen

Die größte Datenbank der Volksrepublik China für wissenschaftliche und technische Informationen ist nach erfolgreichem 18monatigem Test von der Pekinger Stadtverwaltung abgenommen worden. Sie speichert 23000 Daten von 2600 wirtschaftlichen oder wissenschaftlich-technischen Einrichtungen in der VR China. Jede Information soll innerhalb von drei Sekunden abrufbar sein. Die Datenbank ist eine Gemeinschaftsarbeit zwischen dem Pekinger Institut für Wissens- und Technologie-Information mit Betrieben und Forschungeinrichtungen in 20 Provinzen und Städten. ADN

Digitales Simulationssystem DS-88

Das Simulationssystem DS-88 ist zur Bearbeitung von Steuerungs- und Regelungsaufgaben kleineren und mittleren Umfangs vorgesehen und kann in Forschung und Lehre eingesetzt werden. Da bei solchen Aufgaben der Signalflußplan ein übersichtliches und häufig verwendetes Beschreibungsmittel ist, wurde das System blockorientiert aufgebaut. Es erlaubt die Darstellung sowohl kontinuierlicher als auch diskontinuierlicher Übertragungsglieder und die Untersuchung spezieller bei digitalen Regelungen auftretender Effekte (Rundung, Quantisierung).
Das System zeichnet sich durch folgende *Hauptmerkmale* aus:

- hardwaremäßige Voraussetzungen: PC 1715 (vorzugsweise) oder BC A 5120 (lauffähig unter Betriebssystem SCP), Drucker
- flexible Handhabung durch dialogorientierte, menügeführte Bedienung ohne Editier- und Übersetzungsläufe sowie automatische Ermittlung der Rechenfolge (Sortierprogramm)
- Blockvorrat aus 32 Typen von Übertragungsgliedern (erweiterbar)
- Darstellung von linearen kontinuierlichen und diskontinuierlichen Übertragungsgliedern, Nichtlinearitäten, Signalgeneratoren und arithmetischen Gliedern; Integration mit Trapezregel
- Signalflußpläne mit max. 99 Übertragungsgliedern bearbeitbar
- Parametereingaben in Fließpunkt-Dezimaldarstellung mit max. sieben Ziffern
- Ergebnisausgabe als Tabellendruck wahlweise in Fließpunkt-Dezimaldarstellung (vier Ziffern) oder Gleitpunkt-Dezimaldarstellung (sechsstellige Mantisse)
- Möglichkeit der Protokollierung von Eingaben und Zwischenmenüs
- interne Zahlenverarbeitung im 4-Byte-Gleitpunktformat (24 Bit Mantisse)
- Bildschirmgrafik unter Nutzung des zweiten Zeichensatzes mit gleichzeitiger Darstellung von max. drei Kurven (nur PC 1715)
- Pseudografikdruck
- Möglichkeit des Auslagerns bzw. Einlesens von Anwenderprogrammen und Simulationsergebnissen auf bzw. von Diskette.

Bei Interesse wenden Sie sich bitte an die Technische Universität Dresden, Sektion Elektrotechnik, Bereich BAA, Mommsenstraße 13, Dresden, 8027, Tel. 4 63 29 17.

Dittrich/Dr. Müller

Termine

Tagung **Mathematik für die Praxis** mit dem **20. Kolloquium Statistische Qualitätskontrolle** und der **8. Tagung Industriemathematik**

Wer? Technische Hochschule Magdeburg, Sektionen Mathematik und Physik, Rechentechnik und Datenverarbeitung und Technologie der metallverarbeitenden Industrie; FA Statistische Qualitätskontrolle der KDT; Mathematische Gesellschaft der DDR; Bezirksvorstand Magdeburg der KDT
Wann? 7.–9. September 1987
Wo? Technische Hochschule Magdeburg
Was? Aktuelle Probleme und Ergebnisse des Einsatzes von Methoden der Mathematik und der angewandten Informatik in der Praxis, insbesondere in den verschiedenen Bereichen der Technologie, der Vorbereitung des Robotereinsatzes und der Erarbeitung von CAD/CAM-Lösungen.
Daneben ist eine Computergrafik-Ausstellung vorgesehen, für die Arbeiten bis zum 15. 7. 1987 eingereicht werden können.
Wie? Anfragen sind zu richten an die Technische Hochschule „Otto von Guericke" Magdeburg, Sektion Mathematik und Physik, WB Operationsforschung, PSF 124, Magdeburg, 3010, Tel. 59 24 50. *Dr. Schulz*

8. Zuverlässigkeitstagung Qualitätssicherung und Zuverlässigkeitsarbeit für Schlüsseltechnologien

Wer? Fachverband Elektrotechnik der KDT, Wissenschaftliche Sektion Qualitätssicherung und Zuverlässigkeit
Wann? Mai 1987
Wo? Leipzig
Was?
- Theoretische Grundfragen der Zuverlässigkeitstechnik
- Ausfallmechanismen und Ausfallanalyse
- Fertigungsprozeß und Zuverlässigkeit
- Zuverlässigkeit unter Umgebungsbeanspruchungen
- Qualität der Software

Wie? Teilnahmemeldungen schriftlich an: Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Postfach 1315, Berlin, 1086 *Hoppe*

Hinweis

Im Beitrag „Bildschirm-Fensterkopien mit dem KC 85/2(3)" in MP 1/87 auf der 3. Umschlagseite lautet die 8. Zeile von unten in der linken Spalte richtig:
CALL F018 CD18 F0;

Red.

Anwenderzentrum qualifiziert Kunden

Im Anwenderzentrum des VEB Numerik „Karl Marx" wurden im vergangenen Jahr mehr als 5000 Fachleute aus Abnehmerbetrieben qualifiziert. Auf Lehrgängen vermittelte ihnen der Karl-Marx-Städter Betrieb neue Erkenntnisse der Automatisierungstechnik. Die Experten machten sich mit der Programmierung sowie mit Wartung und Instandhaltung von numerischen Steuerungen vertraut. Ausländische Lehrgangsteilnehmer kamen aus der ČSSR, UdSSR, aus Italien, Jugoslawien, Österreich und Polen. ADN

Datenfernübertragung

Das internationale Zentrum für wissenschaftlich-technische Informationen in Moskau bietet den Forschungsinstituten der Akademie der Wissenschaften der UdSSR in Leningrad über eine Direktverbindung jetzt die Möglichkeit, technische Ideen anderer RGW-Mitgliedsländer stärker zu nutzen. Die in den Computern des Zentrums gespeicherten Daten über ausländische Erfindungen können dadurch schneller nach Leningrad übermittelt werden. Beispielsweise werden mathematische Programmpakete der Technischen Universität Dresden für die Steuerung des automatisierten Komplexes des Kalinin-Werkes und die Einführung einer in der DDR entwickelten hocheffektiven Methode zur Schnellbestimmung der Metallschmelze im Leningrader Stahlwalzwerk genutzt. ADN

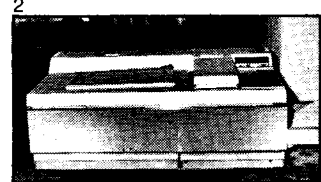
Lizenzen vergeben

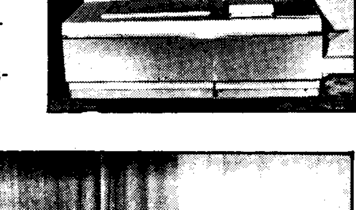
Lizenzverträge auf dem Gebiet der Mikroelektronik hat das Halbleiterwerk Frankfurt/Oder mit Partnern in sozialistischen Ländern Asiens für 1987 abgeschlossen. So erwirbt die Volksrepublik China die Technologie zur Montage eines speziellen Schaltkreises für Tastaturen einschließlich dazu erforderlicher Ausrüstungen. Die Koreanische Demokratische Volksrepublik wird die Herstellung ausgewählter bipolarer Festkörperschaltkreise aufnehmen. Beide Verträge sehen nicht nur die Bereitstellung technischer Dokumentationen, sondern auch die Ausbildung von Spezialisten vor. ADN

Geräte für die grafische Ausgabe

Ende vergangenen Jahres stellte das Unternehmen Rank Xerox Limited in Berlin Erzeugnisse aus seiner Produktionspalette vor. Zum Lieferprogramm des Unternehmens, das mit einem Büro im Internationalen Handelszentrum vertreten ist, gehören in erster Linie Vervielfältigungs- (Kopierer) und Technik für die Bürokommunikation. So waren u. a. zu sehen der Kopierer Xerox 1050, Telekopierer TC 7010 und Schreibmaschine Xerox 6015.
Als kleine, aber leistungsfähige CAD-Systeme wurden der Plotter V-80 (Bild 1) und der Laserprinter Xerox 4045 (Bild 2) in Verbindung mit PC Rank Xerox OP at (Bild 1) vorgestellt. Der OP at ist – nach Informationen des Ausstellers – voll kompatibel mit dem IBM PC AT.
Versatec V-80 arbeitet nach dem elektrostatischen Prinzip; somit sehr geräuscharm. Bis zu 15 Seiten pro Minute kann der Plotter (Drucker) mit Text oder Grafiken bedrucken. Die Auflösung beträgt wahlweise 100 oder 200 Punkte pro Inch. (Siehe auch MP 2/87, Seite 34, Information über den elektrostatischen Zeilendrucker EC 7142 aus der ČSSR.)
Das zweifellos interessanteste Exponat der kleinen Ausstellung war der Laserprinter. Er arbeitet xerographisch, hat eine Druckgeschwindigkeit von bis zu 10 Seiten je Minute. Die Auflösung für Text beträgt 300 × 300 Punkte je Inch. Das Gerät verfügt über maximal 4 Steckplätze für Kassetten, die digitalisierte Schriften enthalten. Dabei können auf jeder Kassette – abhängig von Größe und Umfang der Schrift – bis zu 9 Schriftarten abgespeichert werden. Um Grafik ausgeben zu können, ist eine Speichererweiterung um 384 K DRAM notwendig. Dabei beträgt die Auflösung für eine auszugebene Fläche von 200 mm mal 130 mm Größe (entspricht Papierformat A5) 300 × 300 Punkte je Inch und für eine Flächengröße von 200 mm × 225 mm (entspricht Papierformat A4) 150 mal 150 Punkte je Inch. Bild 3 zeigt einen Screen Dump (Ausschnitt) vom OP at, erstellt mit dem Xerox 4045. MP

2





Messeprogramm Brno 1987

Auf einer Pressekonferenz in Berlin im Januar informierte der Stellvertreter des Generaldirektors des Unternehmens Messen und Ausstellungen Brno (ČSSR), Dr. Jaroslav Kučera, über die diesjährigen Messevorhaben. Neben zahlreichen Fachausstellungen werden vom 9. bis 14. April die 18. Internationale Konsumgütermesse und vom 14. bis 21. September die 29. Internationale Maschinenbaumesse stattfinden. In diesem Jahr wird die Internationale Maschinenbaumesse auf fortschrittliche maschinentechnische Technologien ausgerichtet sein. Auch das wissenschaftlich-technische Rahmenprogramm wird diese Fachproblematik widerspiegeln. MP

Fotos: Paszkowsky (2)

Softwarebanken

In den Kombinaten der DDR werden Bereiche für das Erarbeiten von Software geschaffen bzw. ausgebaut. Weltweit wächst der Wert der Software schneller als der der Hardware, wobei etwa 70 Prozent der benötigten Software für spezifische Lösungen von den Anwendern selbst geschaffen werden. Die Effektivität vieler Schlüsseltechnologien wird maßgeblich durch die Software bestimmt. Die Qualität der Programme ist ein entscheidender Faktor für den wirksamen Einsatz der Rechentechnik und damit für die beschleunigte Steigerung der Arbeitsproduktivität. Gegenwärtig ist der Aufwand für die Hardware und die Software noch etwa gleich groß. Der Trend läßt jedoch erkennen, daß der Aufwand für die Software künftig vierfach größer sein wird als der für die Hardware. Um Doppelarbeit bei der Entwicklung von Software zu vermeiden, werden Softwarebanken geschaffen und das Datennetz ausgebaut. ADN

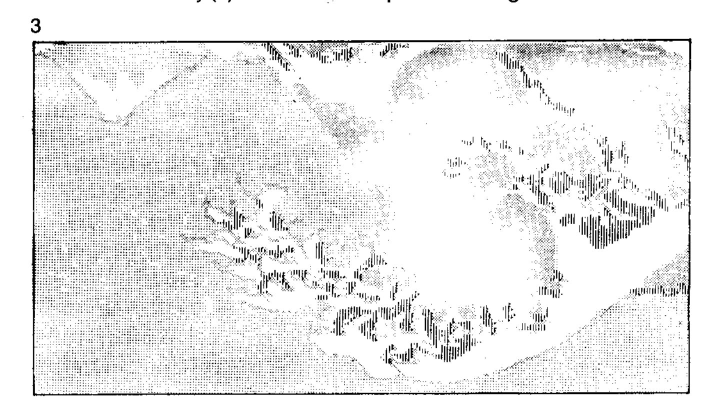
Umwandlung von Bildschirmtexten in Braille-Schrift

Ein Gerät zur Umwandlung von Bildschirmtexten in die abtastbaren Punkte der Braille-Schrift für Blinde ist von Wissenschaftlern der Budapester Technischen Universität konstruiert worden. Im Gegensatz zu einer in den 70er Jahren entwickelten Lösung wird nicht mehr Zeichen für Zeichen, sondern Zeile für Zeile – also 40 und mehr Zeichen auf einmal – transformiert. Als besonders schwierig erwies sich, die Zuverlässigkeit der mechanischen Teile zu garantieren. Beispielsweise müssen 320 Dorne für die Punktschrift mit entsprechender Geschwindigkeit bewegt werden. Das Gerät soll Blinden die Arbeit mit Computern ermöglichen. ADN

1



3



P8000 – ein universelles 16-Bit-Mikrorechnerentwicklungssystem

Dr. Ludwig Claßen
Zentrum für Forschung
und Technologie,
Kombinat VEB Elektro-Apparate-Werke
Berlin-Treptow

0. Einleitung

Das aus einem 8-Bit- und einem 16-Bit-Mikrorechnerteil bestehende P8000 ist ein universell einsetzbares Programmier- und Entwicklungssystem für Multi-User-/Multi-Task-Anwendungen aus dem Kombinat VEB Elektro-Apparate Werke Berlin-Treptow. Die Leistungsfähigkeit jedes Mikrocomputers wird wesentlich durch sein Betriebssoftwaresystem bestimmt; auf dem P8000 sind, um eine große Anzahl von Anwendungsgebieten zu erschließen, vier Betriebssysteme implementiert:

WEGA	(kompatibel UNIX)
UDOS	(kompatibel RIO)
OS/M	(kompatibel CP/M)
IS/M	(kompatibel ISIS II)

Auf dem 16-Bit-Mikrorechnerteil des P8000 läuft das Mehrbenutzer-Betriebssystem WEGA. Es stellt sowohl hinsichtlich seiner Leistungsfähigkeit als auch seiner Anwendungsbreite eine neue Qualität gegenüber bisher bekannten Betriebssystemen für 8-Bit-Mikrorechner dar.
Auf dem 8-Bit-Mikrorechnerteil des P8000 sind, um die Aufwärtskompatibilität zu verfügbaren Softwaresystemen abzusichern, die Betriebssysteme UDOS, OS/M und IS/M implementiert. Dadurch ist die Übernahme vorhandener erprobter Softwarelösungen auf das P8000 problemlos möglich.
Das P8000 bietet durch seine Ausstattung dem Anwender ein breites Spektrum an Softwarearbeitsmöglichkeiten, das für vielfältige Problemlösungen eingesetzt werden kann. Schwerpunkt der vorliegenden Realisierungsversion des P8000-Softwaresystems ist die Unterstützung der Softwareentwicklung für die Mikroprozessorfamilien

- U881/882
  Einchipmikrorechner
- U880
  8-Bit-Mikroprozessorsystem
- U8001/8002
  16-Bit-Mikroprozessorsystem
- K 1810 WM 86
  16-Bit-Mikroprozessorsystem.

Die Einbindung weiterer Mikroprozessorsysteme in das P8000-Entwicklungssystemkonzept ist möglich.
Für Echtzeitaufgaben in Anwendersystemen mit den 16-Bit-Mikroprozessoren U8001/U8002 wird zusätzlich das hocheffektive Real-Time-Betriebssystem IRTS 8000 (Kernel, Monitor, Debugger, Handler usw.) und das zugehörige – auf dem P8000 lauffähige – automatische Generierungssystem ICL 8000 bereitgestellt.

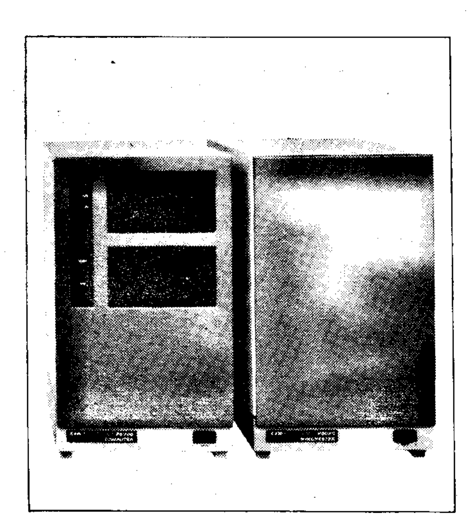


Bild 1 Programmier- und Entwicklungssystem P8000 Foto: DEWAG – Weber

1. Hardwarekonfiguration

Das Gerätesystem P8000 besteht aus mehreren aufeinander abgestimmten Hardwareteilkomponenten /1/, die, abhängig vom jeweiligen Einsatzfall, in verschiedener Weise miteinander konfiguriert werden können (Bild 2):

- *P8000-Grundgerät* mit 8- und 16-Bit-Mikrocomputerzentraleinheit, mit bis zu 1 MByte Hauptspeicher und mit zwei Floppy-Disk-Laufwerken (5¼ Zoll)
- *P8000-Terminalarbeitsplatz* mit alphanumerischem Zeichenvorrat und V.24-Interface
- *P8000-EPROM-Programmiermodul* für EPROM-Schaltkreise der Typen 2700, 2716, 2732 und 2764 (EPROM – erasable programmable read only memory).
- *P8000-Hard-Disk-Beistellgerät* (5¼-Zoll-Winchesterlaufwerk) mit ST506-Anschlußsteuerung für Parallelinterface
- *P8000-Emulatorsystem* für den Einchipmikrorechner U881/U882.

Neben diesen, speziell für das Gerätesystem P8000 vorgesehenen, Hardwarekomponenten können in P8000-Konfigurationen auch beliebige Terminals, Drucker, Rechnerkoppeleinheiten, Grafikarbeitsplätze u. a. m. verwendet werden, die den beim P8000 gegebenen Hardware- und Software-Interfacebedingungen genügen. Die Zentraleinheit des P8000 ist in einem Kompaktgehäuse untergebracht (Abmessungen: 234 × 345 × 400 mm^3). Eine Kartenbaugruppenaufnahme dient innerhalb des P8000-Grundgerätes zur mechanischen Fixierung von zwei miteinander verbundenen Einzelleiterplatten mit dem 8- und 16-Bit-Mikrorechnerteil. Auf der Leiterplatte des 16-Bit-Mikrorechnerteils befinden sich fünf 64polige Steckverbinder, die zur Aufnahme von Speicherbaugruppen mit 64-KBit-DRAM-Speicherschaltkreisen dienen. Auf jeder dieser einzeln steckbaren Speicherbaugruppen ist ein 256-KByte-DRAM-Speicherbereich mit Paritätsfehlerüberwachung untergebracht (DRAM – dynamic random access memory). Die nicht mit Speicherbaugruppen belegten 64poligen Steckverbinder auf dem 16-Bit-Mikrorechnerteil können zur Aufnahme zusätzlicher Ein-/Ausgabeerweiterungsbaugruppen genutzt werden.
Neben den Elektronikbaugruppen befinden sich im P8000-Grundgerät zwei 5¼-Zoll-Floppy-Disk-Laufwerke und eine kompakte Stromversorgungseinheit.
Das P8000-Grundgerät ist standardmäßig mit acht seriellen und zwei parallelen Interfaceschnittstellen (16 Bit) ausgestattet, die zur Ankopplung von Terminalarbeitsplätzen, Hard-Disk-Beistellgeräten, Druckern, In-Circuit-Emulatoren, EPROM-Programmiermodulen, Rechnerkoppeleinheiten u. a. m. dienen können. Im einzelnen hat das P8000-Grundgerät folgende Hardwaremerkmale (Bild 3; siehe auch /1/):

- *8-Bit-Mikrorechnerteil* auf Basis UA880 (4 MHz) mit
  – 2 Fassungen für EPROM-Schaltkreise der Typen 2716, 2732 oder 2764 (4, 8 oder 16 KByte)
  – 8 64-KBit-DRAM-Schaltkreisen U264 (64 KByte)

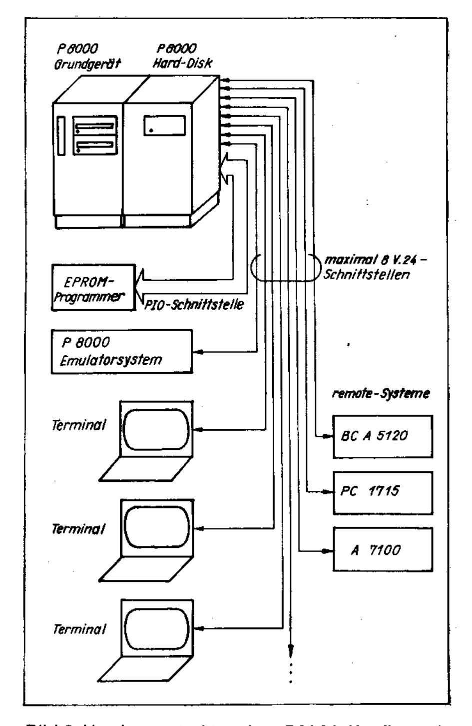


Bild 2 Hardwarestruktur einer P8000-Konfiguration

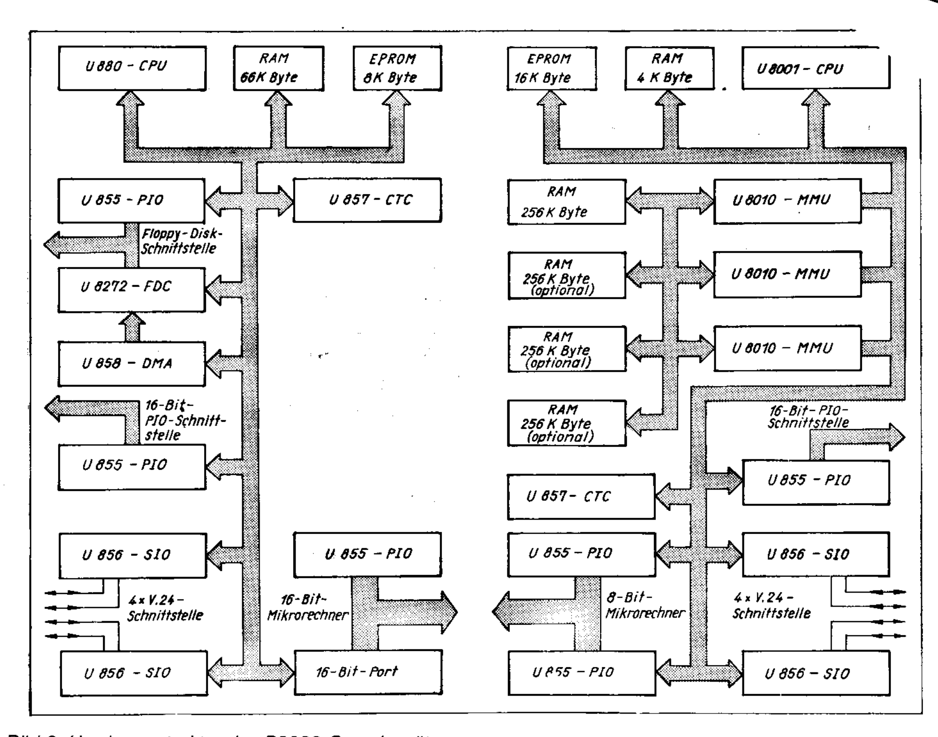


Bild 3 Hardwarestruktur des P8000-Grundgerätes

– 4 1K × 4-SRAM-Schaltkreisen U214 (2 KByte)
– 2 UA856-SIO für vier serielle Schnittstellen (4 × V.24/IFSS, speziell für Terminal- und Druckeranschluß)
– 1 UA855-PIO für zwei 8-Bit-Parallelschnittstellen (speziell für EPROM-Programmiermodulanschluß)
– 1 UA858-DMA für direkten Speicherzugriff
– 1 U8272-FDC für Floppy-Disk-Anschluß (5¼ und 8 Zoll, FM- oder MFM-Aufzeichnungsverfahren (FM single density, MFM double density)

■ *16-Bit-Mikrorechnerteil* auf Basis UB8001 (4 MHz) mit
– 4 Fassungen für EPROM-Schaltkreise der Typen 2716, 2732 oder 2764 (8, 16 oder 32 KByte)
– 4 1K × 4-SRAM-Schaltkreisen U214 (2 KByte)
– 3 UB8010-MMU für Speicherverwaltungsaufgaben
– 2 UA856-SIO für vier serielle Schnittstellen (4 × V.24 oder 2 × V.24 und 2 × IFSS, speziell für Terminal- und Druckeranschluß)
– 1 UA855-PIO für zwei 8-Bit-Parallelschnittstellen (speziell für Hard-Disk-Anschluß)
– 5 Steckverbinder für Speicher- und Ein-/Ausgabeerweiterungskarten (insgesamt bis zu 1 MByte)

■ *Speichererweiterungskarten* mit je 38 64-KBit-DRAM-Schaltkreisen U264 (256 KByte) mit Paritätsfehlerkontrolle

■ zwei in das P8000-Grundgerät integrierte 5¼-Zoll-*Floppy-Disk-Laufwerke* (80 Spuren/FM oder MFM) und Anschlußmöglichkeit für Floppy-Disk-Beistellgerät mit zwei Laufwerken 5¼ Zoll (FM und MFM) oder zwei Laufwerken 8 Zoll (FM und MFM).
Die Kommunikation zwischen dem 8- und 16-Bit-Mikrorechnerteil läuft über einen speziellen 32-Bit-Parallelschnittstellenbus mit begleitenden Handshake-Steuersignalleitungen.
Am P8000-Grundgerät befinden sich neben dem Netzschalter lediglich zwei Bedienelemente, die Taste RESET (Hardware-Reset) und die Taste NMI (NMI-Interrupt).

2. Firmwarekomponenten

Die Softwareteile, die fest und unveränderlich in den EPROM-Speicherbauelementen des P8000-Gerätesystems abgelegt wurden, werden als P8000-Firmware bezeichnet. EPROM-Speicherbauelemente mit Firmware befinden sich im P8000-Grundgerät, im P8000-Terminalarbeitsplatz, im P8000-Hard-Disk-Beistellgerät und im P8000-In-Circuit-Emulator.
Der P8000-EPROM-Programmiermodul arbeitet ohne eigene EPROM-Firmware.
Die im P8000-Grundgerät befindlichen EPROM-Speicherbauelemente umfassen, abhängig vom Bauelementetyp, auf dem 8-Bit-Mikrorechnerteil einen Speicherbereich von 4, 8 oder 16 KByte und auf dem 16-Bit-Mikrorechnerteil einen Speicherbereich von 8, 16 oder 32 KByte.
Im einzelnen enthalten sie die Firmwarekomponenten Hardwareeigentest, U880/U8001-Softwaremonitor, 8-/16-Bit-Mikrorechnerkommunikation und Betriebssystem-Anfangslader.

3. Betriebssoftware des Gerätesystems P8000

3.1. Betriebssystem WEGA

Das Betriebssystem WEGA ist für die unterschiedlichsten Einsatzfälle konzipiert. Es ist ein Mehrbenutzer-Betriebssystem (Multi-User) mit Multitask-Eigenschaften, bei dem jeder Teilnehmer mehrere Prozesse (Programme) gleichzeitig bearbeiten lassen kann. Insgesamt sind bei der WEGA-Implementation auf dem P8000-Grundgerät bis zu acht quasisimultan an einer Zentraleinheit arbeitende Nutzer zulässig. Jeder Nutzer ist über einen Terminalarbeitsplatz mit dem P8000-Grundgerät verbunden.
WEGA ist kompatibel zum UNIX, System III. Es enthält alle Standardmerkmale dieses Betriebssystems, wie hierarchisches Dateiverwaltungssystem, Ein-/Ausgaberedirektion, Pipe- und Filterverarbeitungsmöglichkeiten, Shell-Kommandointerpreter, C-Sprachbasis usw.
Auf Quellkodeebene ist WEGA mit den an UNIX angelehnten Betriebssystemen PSU (ESER), MUTOS (SM 4, K 1600) und zur MUTOS-8000-Implementation auf dem Bürocomputer A 5120.16 kompatibel. Die Nutzung des Betriebssystems WEGA setzt ein an das P8000-Grundgerät über ein PIO-Inter-

face anschließbares Hard-Disk-Beistellgerät (5¼-Zoll-Winchester-Hard-Disk) von minimal 10 MByte voraus. Die maximale Größe des externen Speichermediums ist nahezu unbegrenzt. Als Backup-Medium für den externen Festplatten-Massendatenspeicher werden die im P8000-Grundgerät eingebauten 5¼-Zoll-Floppy-Disk-Laufwerke verwendet. UNIX ist international ein Standardbetriebssystem für 16-/32-Bit-Mikrorechner /2...5/. Es wird auf unterschiedlichsten Rechnern, vom Mikro- über Klein- bis zu den Großrechnern, vom Arbeitsplatzcomputer über CAD/CAM-Systeme bis zu EDV-Anlagen, eingesetzt. Entsprechend umfangreich ist die verfügbare Software.
Das P8000-Softwaresystem ist in der vorliegenden Realisierungsversion primär vorgesehen als Entwicklungssystem zur Programmerstellung für verschiedene Mikroprozessorfamilien und als Zentralgerät zum Anschluß des P8000-Emulators.

Das P8000 ist darüber hinaus aber auch sofort ohne Zusatz oder Änderung einsetzbar für

■ Aufgaben bei der Rationalisierung und Automatisierung der Büro- und Verwaltungsarbeit
■ Basiskonfigurationen beim Aufbau von komplexen, allen Anforderungen entsprechenden Datenbanksystemen
■ Textverarbeitungsaufgaben bis hin zur Aufbereitung von Texten für den Lichtsatz
■ Zentralrechnersysteme, die dezentrale Echtzeitsteuer- und -regelsysteme überwachen und Dateiarbeit ausführen können
■ Rechnerkopplungen des P8000-Grundgerätes mit gleichartigen oder mit unterschiedlichen Rechnern, wie Bürocomputern und Kleinrechnern sowie für Kopplungen des P8000-Grundgerätes über ein Modem an ein Datennetz
■ Unterstützungsaufgaben bei der Entwicklung von Compilern für spezielle Sprachen (Fachsprachen der Meß-, Steuer- und Regelungstechnik, spezielle Prüfautomatensprachen usw.) durch Compiler-Entwicklungssoftware und für vieles andere mehr.

3.1.1. Struktur des Betriebssystems WEGA

Das WEGA-Softwaresystem des P8000 besteht aus folgenden Komponenten:

■ Anfangsladerprogramm und Stand-alone-Programme, die unter Steuerung des Anfangsladers arbeiten
■ WEGA-Betriebssystemkern
■ System-, Dienst- und Anwenderprogramme, die unter Steuerung des Betriebssystems WEGA arbeiten.

Nach dem Netzeinschalten und nach jedem Hardware-RESET ist die Software im EPROM-Firmwarespeicher des P8000 aktiv. In ihr sind Debug-Funktionen des U880-/U8001-Softwaremonitors für die maschinennahe Softwarefehlersuche, ein Hardwareeigentestprogramm, Programme zur Abwicklung der Kommunikation zwischen 8- und 16-Bit-Mikrorechnern im P8000 sowie ein Programm zur automatischen Initialisierung des WEGA-Anfangsladers enthalten.
Der Ladeprozeß von WEGA erfolgt grundsätzlich von der Festplatte. Über die im P8000-Grundgerät eingebauten Floppy-Disk-Laufwerke kann die Festplatte bei Bedarf mit Hilfe der Stand-alone-Programme neu eingerichtet werden.
Das Betriebssystem WEGA kennt zwei Betriebsarten, den Single-User-Mode und den Multi-User-Mode.
Im Single-User-Mode ist nur die V.24-Systemkonsole des WEGA-Systemprogrammierers (WEGA-Superuser) aktiv. In dieser Betriebsart wird die Softwarewartung des Systems durchgeführt. Diese Betriebsart kann auch gewählt werden, wenn das P8000 als Einbenutzer-System eingesetzt werden soll.
Im Multi-User-Mode werden alle angeschlossenen und eingeschalteten Terminals aktiviert. Bis zu acht Nutzer können über separate Terminalarbeitsplätze in dieser Betriebsart quasisimultan an einem P8000-Grundgerät unter WEGA arbeiten.

Folgende Ein-/Ausgabegeräte sind standardmäßig unter Steuerung des WEGA-Betriebssystemkerns verfügbar:

■ 8 serielle V.24-Kanäle für die WEGA-Systemkonsole, für Anwenderterminals, für einen Drucker (Typ: robotron K631x mit V.24-Schnittstelle) und zur Kopplung des P8000 mit weiteren Rechnern
■ 2 Festplattenlaufwerke (P8000-Hard-Disk-Beistellgerät) mit jeweils mehreren logischen Dateisystemen, angeschlossen über einen ST506-Hard-Disk-Controller mit Parallelinterface
■ 2 Floppy-Disk-Laufwerke 5¼ Zoll (MFM-Aufzeichnungsverfahren, 80 Spuren) als Back-Up-Medium.
Neben diesen Standardgeräten kann der Anwender, seinen speziellen Erfordernissen entsprechend, zusätzliche Ein-/Ausgabegeräte in das WEGA-Systemkonzept über spezielle Treiberprogramme einbinden.

3.1.2. Stand-alone-, System- und Dienstprogramme WEGA

Die Stand-alone, System- und Dienstprogramme des Betriebssystems WEGA bilden die Softwarebasis des P8000-Anwenders. Sie unterstützen ihn bei der Lösung seiner speziellen Aufgabenstellung in einem breiten Bereich.

● *Stand-alone-Programme*

Die Stand-alone-Programme dienen zur Initialisierung des WEGA-Softwaresystems. Ihre Nutzung ist nur beim Neustart, bei totalen Systemzusammenbrüchen und bei Softwarewartungsarbeiten notwendig. Die Stand-alone-Programme arbeiten unter Steuerung des WEGA-Anfangsladeprogrammes.
boot
Anfangslader (boot1)
format
Formatierungsprogramm Hard-Disk
mkfs
Einrichten eines WEGA-Dateisystems
install
Laden des Betriebssystems WEGA

● *Systemprogramme*

Die WEGA-Systemprogramme bilden ein Grundbausteinsortiment, das der Anwender standardmäßig bei der Lieferung des Betriebssystems WEGA komplett übergeben bekommt (WEGA-Kommandosatz). Die Systemprogramme überstreichen breite Anwendungsbereiche, von der Rechnerkommunikation über die Textverarbeitung bis zur Softwareentwicklungsunterstützung. Sie sichern außerdem den Zugriff des Anwenders zu den gesamten WEGA-Betriebssystemressourcen.
Zum WEGA-Standardsoftwarekommandosatz gehören insbesondere die in der Tafel 1 (auf der 2. Umschlagseite) wiedergegebenen Systemprogramme. Weitere Systemprogramme, wie ein nutzerbezogenes Rechenzeitabrechnungsprogramm, ein Fortschreibungssystem für Anwenderquellprogramme (sccs-source code control system), ein Utilitiessystem für die indexsequentielle Behandlung von Dateien, Koppelprogramme für den BC A 5120 bzw. den PC 1715 u. a. m., können, in Abhängigkeit von der Aufgabenstellung und der Größe des verfügbaren Hard-Disk-Plattenspeichers, genutzt werden.

● *Dienstprogramme*

Zur Abarbeitung unter Steuerung des WEGA-Softwaresystems werden Dienstprogrammpakete für die Mikroprozessorsoftwareentwicklung, für Datenbankverwaltungssysteme und für vieles andere mehr bereitgestellt.

Insbesondere stehen folgende Softwarepakete auf dem P8000-Gerätesystem unter Steuerung des Betriebssystems WEGA zur Verfügung:

■ U8001/U8002 Assemblerprogrammentwicklungspaket
■ K 1810 WM 86 Assemblerprogrammentwicklungspaket

■ U880 Assemblerprogrammentwicklungspaket (mit C-Compiler)
■ U881/U882 Assemblerprogrammentwicklungspaket
■ Compilersystem für die höheren Programmiersprachen C, PASCAL, FORTRAN 77, BASIC, PLZ/SYS
■ Relationales Datenbankverwaltungssystem WEGA-DATA
■ Planungs- und Modellierungsprogrammsystem WEGA-CALC
■ Textverarbeitungsprogrammsystem WEGA-TEXT
■ Grafiksoftwaresystem WEGA-GRAFIK/IGE
■ Emulationsprogrammsystem WEGA-SCP-EMULATOR
■ Echtzeitsoftwaregenerierungssystem IRTS/ICL-8001/8002

3.2. Betriebssystem UDOS

Das Betriebssystem UDOS ist ein Floppy-Disk-orientiertes Betriebssystem für den 8-Bit-Mikrorechnerteil des P8000 /6/, /7/. Es zeichnet sich durch ein komfortables Dateiverwaltungssystem, die wahlfreie Zuordnung von Ein-/Ausgabedatenströmen, die automatische Speicherplatzverwaltung und einen umfangreichen Kommandosatz aus.
Unter Steuerung des Betriebssystems UDOS ist insbesondere Entwicklungssoftware für die in der DDR produzierten Mikroprozessorfamilien U880, U881/U882 und U8001/U8002 verfügbar. Daneben existiert eine Fülle weiterer UDOS-Softwarepakete, von Compilern für höhere Programmiersprachen über Makroprozessoren bis hin zur Textverarbeitung.
Das auf dem 8-Bit-Mikrorechnerteil des P8000 laufende UDOS ist mit dem für die Personal- und Bürocomputer PC 1715 bzw. A 5120 existierenden UDOS 1715/U-1526 im grundsätzlichen Aufbau identisch. Zum UDOS 1715 existiert beim P8000 zusätzlich eine absolute Diskettenkompatibilität.

3.3. Betriebssystem OS/M

Das Betriebssystem OS/M ist ein Floppy-Disk-orientiertes Betriebssystem für den 8-Bit-Mikrorechnerteil des P8000 /8/, /9/. Die Anwendung von OS/M ist begründet in der Verfügbarkeit einer großen Anzahl von Dienstprogrammen, von Compilern/Interpretern für höhere Programmiersprachen und von Applikationssoftwaresystemen. Sie betreffen den Komplex der kommerziellen Datenverarbeitung, den Komplex der Mikrorechnerentwicklungssoftware, aber auch viele weitere Anwendungslinien. Die Schnittstelle von OS/M auf dem P8000 zu den unter seiner Steuerung laufenden Programmen ist voll kompatibel mit der Schnittstelle des Betriebssystems CP/M, Version 2.2.
Außerdem ist beim P8000 die Kompatibilität mit dem in der DDR verfügbaren Betriebssystem SCPX 1526 für die Bürocomputer A 5120/A 5130 und SCPX 1715 für den Personalcomputer PC 1715 gegeben. Zum SCPX 1715 existiert beim P8000 zusätzlich eine absolute Diskettenkompatibilität.
Mit zwei optional an das P8000 anschließbaren 8-Zoll-Standardlaufwerken wird die Dateiübernahme von CP/M-Standarddisketten einfacher Aufzeichnungsdichte (entsprechend dem Standardformat 3740) gewährleistet.

3.4. Betriebssystem IS/M

Das Floppy-Disk-Betriebssystem IS/M für den 8-Bit-Mikrorechnerteil des P8000 unterstützt vorrangig die Programmentwicklung für die 16-Bit-Mikroprozessorlinie K 1810 WM 86. Die Schnittstelle IS/M auf dem P8000 ist vollständig kompatibel mit der Schnittstelle des Betriebssystems ISIS II. Die Kompatibilität sichert auf dem P8000 die Lauffähigkeit aller ISIS-II-Programme auf Maschinenkodeebene.
Mit zwei an das P8000 anschließbaren 8-Zoll-Standardlaufwerken wird die Diskettenkompatibilität zu originalen ISIS-II-Standarddisketten einfacher Aufzeichnungsdichte (entsprechend Standardformat 3740) erreicht.
Bei 5¼-Zoll-Minidisketten existiert für ISIS II international kein standardisiertes Format. Auf den 5¼-Zoll-Laufwerken des P8000 wird deshalb ein Format analog dem UDOS- und OS/M-Format mit 80 Spuren (Standardformat 34) realisiert.

Literatur

/1/ Zuchold, W.; Heller, M.: Hardwarekonfiguration P8000. Firmenschrift KEAW; 1986
/2/ UNIX (TM) User's Manual Release 3.0. Firmenschrift Bell Inc.; Juni 1980
/3/ Programmers Manual for UNIX System III, Band 2A/2B. Firmenschrift WEC Inc.; Oktober 1981
/4/ Claßen, L.; Oefler, U.: UNIX und C – Ein Anwenderhandbuch. VEB Verlag Technik 1987 (In Vorbereitung)
/5/ Banahan, M.; Rutter, A.: UNIX: Lernen, verstehen, anwenden. Carl Hanser Verlag; München Wien 1984
/6/ Claßen, L; Oefler, U.: Wissensspeicher Mikrorechnerprogrammierung. VEB Verlag Technik 1986
/7/ Claßen, L.; Brennenstuhl, H.; Gröger, R.: Universelles Software-Entwicklungskonzept UNOS/GDS 6000. Radio, Ferns., Elektron., Berlin 32 (1983) 8, S. 483–485

Fortsetzung auf Seite 76

Parallelverarbeitende Rechnersysteme

Dr. Hans Heuer
Technische Hochschule Magdeburg, Sektion Rechentechnik und Datenverarbeitung

Die Fortschritte bei der Erhöhung der Rechnerleistung waren in der Vergangenheit hauptsächlich auf Verbesserungen in der Bauelemente-Technologie zurückzuführen. Hier nähert man sich jedoch inzwischen einer physikalischen bzw. technischen Grenze. Insbesondere gilt dies für die wichtige technische Kennziffer der Operationsgeschwindigkeit, die zwar nicht das einzige Maß für die Leistungsfähigkeit eines Rechnersystems darstellt, jedoch als ein fundamentales Merkmal angesehen werden muß.
Ein weiterer spürbarer Zuwachs der Verarbeitungsgeschwindigkeit kann nur auf der Basis innovativer Rechnerarchitekturen erzielt werden. Diese sind im Unterschied zu der streng sequentiellen Befehlsabarbeitung der konventionellen von-Neumann-Rechner durch eine parallele Verarbeitung gekennzeichnet.

Einschätzung des gegenwärtigen Standes

In den letzten beiden Jahrzehnten wurden in internationalem Maßstab bereits Rechnersysteme mit teilweiser oder vollständiger Parallelverarbeitung konzipiert bzw. hergestellt und genutzt.
Nach Flynn werden die Rechnersysteme entsprechend der Anzahl der Befehls- und Datenströme in vier Klassen eingeteilt /2/, /3/:

SISD	Single Instruction – Single Data
SIMD	Single Instruction – Multiple Data
MISD	Multiple Instruction – Single Data
MIMD	Multiple Instruction – Multiple Data.

Die Klasse **SISD** umfaßt die gewöhnlichen Einprozessorsysteme (EPS) als Abkömmlinge des von-Neumann-Konzepts. In der Praxis sind reine von-Neumann-Maschinen selten anzutreffen, weil immer ein gewisser Grad von Parallelismus notwendig ist, um eine hohe Verarbeitungsgeschwindigkeit zu erreichen.
Zu den Systemen der Klasse **SIMD** gehören die auch als Arrayrechner

bezeichneten Feldrechner sowie die Assoziativrechner. Feldrechner besitzen ein Feld (Array) von identischen Verarbeitungselementen. Sie realisieren eine synchrone parallele Verarbeitung, das heißt, sie realisieren in gleichen Zeitabschnitten die gleichen Befehle. Aufbau und Wirkungsweise der Assoziativrechner sind im wesentlichen durch assoziative Suchprozesse bestimmt.
In die Klasse **MISD** können die Pipeline- oder Fließbandrechner eingereiht werden. Diese verfügen über einen Prozessor, der aufgrund einer entsprechenden Unterteilung der Befehle in jeweils mehrere Stufen die gleichzeitige Ausführung entsprechend vieler aufeinanderfolgender Befehle in unterschiedlichen Stufen und ihre Weitergabe an die nächste Stufe vornimmt.
Die Klasse **MIMD** enthält die auch kurz als Multiprozessoren bezeichneten Multiprozessor- oder Mehrprozessorsysteme (MPS). Dies sind Rechnersysteme, die mehrere selbständige Prozessoren (SI-Prozessoren) besitzen, die gleichzeitig und unabhängig voneinander an der Lösung einer Aufgabe arbeiten. Sie realisieren eine asynchrone parallele Verarbeitung. Falls eine Parallelisierung der Berechnungen für eine einzelne Aufgabe Schwierigkeiten bereitet, kann auch durch gleichzeitige Ausführung unterschiedlicher Aufgaben auf den einzelnen Prozessoren die Auslastung der Prozessoren erreicht werden. Von allen parallelverarbeitenden Systemen besitzen die Multiprozessorsysteme den höchsten Grad der Universalität. Daher sollte ihnen ganz besondere Aufmerksamkeit gewidmet werden.
Durch die Entwicklung der Mikroprozessortechnik, die die Bereitstellung leistungsfähiger Prozessoren zu niedrigen Preisen ermöglicht, ist es heute prinzipiell möglich, Multiprozessorsysteme mit kostengünstigen Komponenten zu realisieren. In der jüngsten Vergangenheit war man noch auf die aufwendigeren Kleinrechner angewiesen. Dadurch bedingt war auch eine Beschränkung auf eine relativ kleine Anzahl von Prozessoren. Hier ist das klassische Multiprozessorsystem C.mmp, der Carnegie-Mellon-multi-miniprocessor, zu nennen /1/, /4/. Es verfügte über 16 Prozessoren. Dies waren verschiedene Modelle der DEC PDP-11. Jedem Prozessor war ein privater (lokaler) Speicher zugeordnet. Jeder der 16 Prozessoren konnte weiterhin über ein 16 × 16-Crossbar-Verbindungsnetzwerk mit jedem der 16 Moduln des gemeinsamen (globalen) Speichers verbunden werden.
Auf der Basis der mit diesem System gewonnenen Erfahrungen wurde an der Carnegie-Mellon-Universität ein

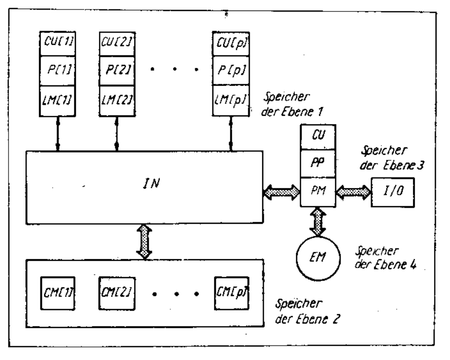


Bild 1 Struktur eines universellen und flexiblen Multiprozessorsystems

System mit einer veränderten Struktur entwickelt und gebaut. Dieses unter dem Namen Cm∗ bekannt gewordene System zeichnet sich durch eine hierarchische Struktur aus. Sein Grundbaustein ist das als Computer-Modul (Cm) bezeichnete Paar Prozessor – Speicher. Bis zu 14 Computer-Moduln können über einen Bus zu einem Cluster verbunden werden. Mehrere Cluster wiederum können über einen Intercluster-Bus gekoppelt werden. Die Kosten für den Zugriff zu einem lokalen Speicher eines Moduls, zu einem Modul im gleichen Cluster bzw. zu einem Modul in einem anderen Cluster verhalten sich wie 1:3:9. Eine wesentliche Voraussetzung für eine effektive Arbeit des Systems ist daher die Verfügbarkeit der Mehrzahl der Programme und Daten in dem dem entsprechenden Prozessor zugeordneten Speicher.
Als Prototyp eines Multiprozessorsystems mit einer größeren Anzahl von Mikroprozessoren ist das Rechnersystem SMS 201 (Structured Multi-microprocessor System 201) der Firma Siemens bekannt geworden /11/. Es besitzt 128 Mikroprozessoren, von denen je 16 zu einem Block zusammengefaßt wurden, sowie einen Steuerrechner. Das System wurde ursprünglich für die Simulation von „parallelen“ Problemen, wie Wetterprognose, Potentialfelder, Mustererkennung und Verfahrensuntersuchungen der numerischen Mathematik, entworfen. Jedem Punkt des behandelten Raumes sollte ein Mikroprozessor entsprechen. Programme und Daten sind in den privaten Speichern der Prozessoren untergebracht. Damit können alle Prozessoren gleichzeitig und unabhängig voneinander arbeiten. Anschließend werden die Ergebnisse der Berechnung unter den Prozessoren ausgetauscht, die Berechnungen werden erneut ausgeführt und so weiter. So erfolgt eine ständige Wiederholung des aus der Rechenphase und der Datenaustauschphase bestehenden Zyklus.
An diesem Rechnersystem wird deutlich, daß durch die moderne Mikroprozessortechnik wesentliche Voraussetzungen für die Entwicklung von Rechnersystemen mit Parallelverarbeitung geschaffen wurden. Aufgrund der niedrigen Herstellungskosten für integrierte Schaltungen ist es sinnvoll, eine noch größere Anzahl von Mikroprozessoren für die Lösung einer einzelnen Aufgabe einzusetzen und so Rechnersysteme mit noch höheren Leistungen zu entwickeln. Allgemein gilt, daß bei gegebener Leistung der Prozessoren die Gesamtleistung eines Multiprozessorsystems um so größer ist, je mehr Prozessoren es enthält. Für die exakte Bewertung und den Vergleich eines Multiprozessorsystems gegenüber einem entsprechenden Einprozessorsystem werden als wesentliche Kennziffern die Beschleunigung S_p und der Wirkungsgrad E_p benutzt.
Diese sind durch die Formeln

$$S_p = \frac{T_1}{T_p}$$

$$E_p = \frac{S_p}{p}$$

bestimmt. Hierbei ist T_p die Zeit für die Lösung einer Aufgabe auf einem Multiprozessorsystem mit p Prozessoren, während T_1 die Zeit für die Lösung einer Aufgabe desselben Typs auf einem Einprozessorsystem darstellt. Die Beschleunigung kann höchstens den Wert p annehmen. Im allgemeinen ist sie jedoch kleiner.
Bezüglich der möglichen Erhöhung der Verarbeitungsgeschwindigkeit in Abhängigkeit von der Anzahl der eingesetzten Verarbeitungselemente gibt es unterschiedliche Aussagen. Die Diskussion wird insbesondere durch die Behauptung von Minsky beeinflußt, die in der Fachwelt die Bezeichnung „Minskysche Vermutung“ erhalten hat /5/, /6/. Danach ist für Rechnerarchitekturen mit p-fachem Datenstrom die erreichbare Beschleunigung nicht proportional zu p, sondern nur zu $\log_2 p$. Dies würde bedeuten, daß für eine Erhöhung der Leistung eines Rechnersystems mit dem Faktor 10 bereits 1024 Prozessoren benötigt würden. Diese Theorie konnte bisher nicht entscheidend widerlegt werden, so daß nach wie vor Zweifel an der Effektivität der Parallelverarbeitung bestehen.
Auch bei wohlwollender Beurteilung des gegenwärtigen Standes muß eingeschätzt werden, daß die den Multiprozessorsystemen eigenen Vorteile und Möglichkeiten bisher noch in keiner Weise zur Wirkung gebracht worden sind. Um zu realistischen Aussagen über das Leistungsverhalten der innovativen Rechnerarchitekturen zu gelangen, ist es notwendig, die drei Komponenten Hardware, Software und Algorithmen (Brainware) als eine untrennbare Ein-

heit zu sehen. Das grundlegende Problem besteht darin, eine solche Struktur eines Rechnersystems zu bestimmen, die so universell und ökonomisch ist, daß sie die Lösung einer breiten Klasse von Aufgaben mit hoher Effektivität ermöglicht. Das bedeutet, daß das Rechnersystem u. a. folgende Eigenschaften besitzen muß:

- Modularität
- Veränderbarkeit der Struktur
- Existenz einer Speicherhierarchie
- kostengünstige Komponenten
- Proportionalität der Verarbeitungsgeschwindigkeit zur Anzahl der Prozessoren.

Von dem sowjetischen Wissenschaftler Moltschanow vom Institut für Kybernetik der Ukrainischen Akademie der Wissenschaften Kiew wurden für derartige Rechnersysteme effektive parallele Algorithmen entwickelt, die eine hohe Beschleunigung gewährleisten /7/ bis /10/. Vom Autor wurde diese Linie weitergeführt, indem für die Lösung wichtiger Klassen wissenschaftlich-technischer Aufgaben ein Modell eines universellen Multiprozessorsystems definiert wurde, das eine adäquate Widerspiegelung der für die gestellte Aufgabe wesentlichen Seiten des Untersuchungsgegenstandes ermöglicht, und auf dessen Grundlage effektive parallele Algorithmen entwickelt wurden.

Ein universelles und flexibles Multiprozessorsystem

Zur Demonstration der Leistungsfähigkeit der Parallelverarbeitung wurde ein geeignetes Rechnersystem mit mehrfachem Befehls- und Datenstrom mit der in Bild 1 angegebenen Struktur konzipiert. Das System besitzt p ($p \geqq 2$) identische Prozessoren, wobei der Wert von p eine Potenz der Zahl 2 mit positivem ganzzahligem Exponenten darstellt. Diese Prozessoren realisieren die Verarbeitung. Ein peripherer Prozessor dient der Ein- und Ausgabe.
Weiterhin verfügt dieses Multiprozessorsystem über eine Speicherhierarchie, die aus folgenden Ebenen besteht:

Ebene 1 – lokale Speicher der Prozessoren
Ebene 2 – gemeinsamer Speicher
Ebene 3 – Arbeitsspeicher des peripheren Prozessors
Ebene 4 – Externspeicher des peripheren Prozessors.

Über ein Verbindungsnetzwerk können alle Prozessoren paarweise miteinander sowie mit dem gemeinsamen Speicher verbunden werden. Der gemeinsame Speicher besteht aus p Moduln, von denen jedem Prozessor eine bestimmte Anzahl dynamisch zugeordnet werden kann. Über das Verbindungsnetzwerk können die Prozessoren bzw. die Moduln des gemeinsamen Speichers außerdem mit dem peripheren Prozessor verbunden werden.
Für die Kennzeichnung des Rechnersystems werden folgende Parameter herangezogen:

p	Anzahl der Prozessoren
t	Zeit für die Ausführung einer arithmetischen Operation durch einen Prozessor
t_{S11}	Zeit für den Start eines Transfers zwischen zwei Speichern der Ebene 1
t_{T11}	Zeit für den Transfer eines Wortes zwischen zwei Speichern der Ebene 1
t_{S12}	Zeit für den Start eines Transfers zwischen einem Speicher der Ebene 1 und einem Speicher der Ebene 2 oder umgekehrt
t_{T12}	Zeit für den Transfer eines Wortes zwischen einem Speicher der Ebene 1 und einem Speicher der Ebene 2 oder umgekehrt
M_1	Kapazität in Worten des Speichers der Ebene 1 eines entsprechenden Einprozessorsystems, die für die Aufbewahrung von Daten zur Verfügung steht
M_p	Kapazität in Worten eines lokalen Speichers der Ebene 1 des Multiprozessorsystems, die für die Aufbewahrung von Daten zur Verfügung steht. Ohne Einschränkung der Allgemeinheit sei $M_1 = pM_p$.

Entsprechend der Forderung, daß für jede Klasse von Rechnersystemen mit paralleler Verarbeitung speziell angepaßte Software bzw. Algorithmen zu entwickeln sind, um die Leistung des Systems optimal zu nutzen, erfolgte die Entwicklung einer Reihe numerischer Methoden. Diese gewährleisten eine hohe Beschleunigung und einen hohen Wirkungsgrad.
So ergeben sich zum Beispiel für die parallele Version des bekannten Gaußschen Algorithmus die Beschleunigung und der Wirkungsgrad aus den Formeln (1) und (2).

$$S_p = \frac{p\left\{\frac{2}{3}n^3t + \frac{2n^4}{M_1^2}\left(t_{S12} + \frac{M_1}{2}t_{T12}\right)\right\}}{\frac{2}{3}n^3t + \frac{2n^4}{(M_p-n)^2}\left(t_{S12} + \frac{M_p-n}{2}t_{T12}\right) + (\log_2 p)\left(t_{S11} + \frac{M_p-n}{2}t_{T11}\right) + p(\log_2 p)nt_{S11}} \quad (1)$$

$$E_p = \frac{\frac{2}{3}n^3t + \frac{2n^4}{M_1^2}\left(t_{S12} + \frac{M_1}{2}t_{T12}\right)}{\frac{2}{3}n^3t + \frac{2n^4}{(M_p-n)^2}\left(t_{S12} + \frac{M_p-n}{2}t_{T12}\right) + (\log_2 p)\left(t_{S11} + \frac{M_p-n}{2}t_{T11}\right) + p(\log_2 p)nt_{S11}} \quad (2)$$

Hier bedeutet n die Ordnung des linearen algebraischen Gleichungssystems ($n \gg 1$).
Diese Formeln zeigen, daß für das konzipierte Multiprozessorsystem unter der Voraussetzung, daß die technischen Daten geeignet gewählt werden, die Beschleunigung annähernd proportional zur Anzahl der Prozessoren ist. Dies gilt nicht nur für die parallele Realisierung des Gaußschen Algorithmus, sondern für eine Vielzahl weiterer numerischer Methoden.

Bewertung und Ausblick

Die vom Autor auf der Basis des universellen und flexiblen Multiprozessorsystems gewonnenen Ergebnisse widerlegen die Minskysche Vermutung und beweisen die Zweckmäßigkeit des Einsatzes von Rechnersystemen mit Parallelverarbeitung zur Beschleunigung der Verarbeitung. Auf Grund seiner Struktur kann dieser Multiprozessor mit vertretbarem Aufwand unter Verwendung von Komponenten der modernen Mikroprozessortechnik realisiert werden.
Dies ermöglicht Experimente zum detaillierten Untersuchen der Einflüsse der einzelnen Komponenten des Systems auf die Gesamtleistung. Damit ergeben sich wichtige Hinweise für die weitere Leistungserhöhung.
Für die Zukunft zeichnet sich ab, daß die Bedeutung der Multiprozessorsysteme, die gegenüber allen anderen Architekturen und Strukturen den universellsten Charakter besitzen und die potentiellen Möglichkeiten zur Erzielung einer hohen Beschleunigung und eines hohen Wirkungsgrades der Parallelverarbeitung bieten, weiter zunehmen wird. Insbesondere die flexibleren Multiprozessoren mit Umkonfigurationskonzepten werden im Verlauf der weiteren Entwicklung sicherlich eine wesentliche Rolle spielen und damit stärker als bisher in das Blickfeld der Informatik rücken.

Literatur

/1/ Enslow, Philip, H.: Multiprocessor Organization – A Survey. Computing Surveys 9 (1977) 1, S. 103–129
/2/ Flynn, M. J.: Very high speed computing systems. Proc. IEEE 54 (1966) 12, S. 1901 bis 1909
/3/ Flynn, M. J.: Some Computer Organizations and their Effectiveness. IEEE Transactions on Computers C-21 (1972) 9, S. 948–960
/4/ Haynes, Leonhard S. u. a.: A Survey of Highly Parallel Computing. Computer (1982), S. 9 bis 24
/5/ Minsky, M: Form and content in computer science. ACM Turing Lecture. Journal of the ACM 17 (1970) 2, S. 197–215
/6/ Minsky, M.; Papert, S.: On some associative, parallel and analog computations. In: Jacks, E. L. (edit.): Associative Information Techniques, Elsevier, New York/London/Amsterdam 1971, S. 27–47
/7/ Molčanov, I. N.; Galba, E. F.; Stepanec, N. I.: Realizacija iteracionnych metodov na mnogoprocessornoj ĖVM s parallelnoj organizaciej vyčislenij. Kibernetika (1983) 4, S. 18–23 (Russ.)

Fortsetzung auf Seite 74

Lokale Rechnernetze mit OSI-Architektur

Prof. Dr. Volker Heymer
Akademie der Wissenschaften der DDR
Institut für Informatik
und Rechentechnik

Durch den massenhaften Einsatz von Rechnersystemen unterschiedlicher Leistung (vor allem von Personalcomputern) und die mit ihrer Hilfe arbeitsteilige Bearbeitung komplexer Aufgaben entstand die Notwendigkeit der Verkopplung dieser Systeme. Da für viele Anwendungen die Kommunikation in lokal begrenzten Bereichen abläuft, bildeten sich als adäquate Lösung lokale Netze von Rechnersystemen (Local Area Network – LAN) /1, 2, 3/ heraus, die charakterisiert sind durch eine räumliche Ausdehnung von 100 m bis 10 km sowie die Nutzung eines relativ zuverlässigen Übertragungsmediums mit einem Datendurchsatz größer 100 KBit/s.
Wichtige Anwendungsgebiete der lokalen Netze sind:

– Konstruktion und technologische Produktionsvorbereitung (CAD), Produktionsautomatisierung (CAM), rechnerintegrierte Produktion (CIM)
– Automatisieren der Prozeßsteuerung, Meßwerterfassung, Überwachung
– Büroautomatisierung, Leitung, Verwaltung
– Informationsaustausch in Rechenzentren
– Forschung und Entwicklung
– Ausbildung, Lehre und Schulbildung.

Die Vernetzung der vorher isoliert genutzten Rechnersysteme führt zu spürbaren Effektivitätsgewinnen, die sich ergeben aus
– dem Zugriff auf Rechenleistungen, die über die Leistung von Personalcomputern (PC) am Arbeitsplatz hinausgehen
– der Verbesserung des Informationsflusses, insbesondere bei der Produktionsvorbereitung und -durchführung
– der effektiven Pflege und Nutzung zentraler Datenbanken
– der Einsparung lokaler Ressourcen durch globale Bereitstellung (kollektive Nutzung hochwertiger Peripheriegeräte wie Plotter, Hochleistungsdrucker, Massenspeicher)
– der Einsparung von Wege- und Arbeitszeit und Transportkosten
– der Rationalisierung des innerbetrieblichen Schriftverkehrs (80 Prozent des Gesamtschriftverkehrs)
– Integrationsmöglichkeiten für neue Kommunikationsdienste (Sprach- und Bildkommunikation)
– der Einsparung von Verkabelung und Leitungskonzentratoren
– der Möglichkeit, kurzfristig hohe Übertragungskapazitäten bereitzustellen
– leichter beherrschbaren Systemstrukturen
– der höheren Modularität und besseren Erweiterbarkeit
– dem höheren Standardisierungsgrad.

Dabei wird die Effektivität der CAD/CAM- und CIM-Lösungen durch Vernetzung über lokale Netze um etwa 30 Prozent erhöht.

Kommunikationsarchitektur lokaler Netze

LAN-Hauptkomponenten

Auf der Basis der in den vergangenen Jahren in großer Typen-Vielfalt entstandenen LAN wurde die in Bild 1 gezeigte Struktur entwickelt, die folgende Hauptkomponenten enthält:

● das Übertragungsmedium in Form von Koaxialkabel, verdrillten Leitungen oder Lichtwellenleitern
● die Kabelanschlußeinheiten MAU (Medium Attachment Unit) für die Kopplung des Übertragungsmediums an die Netzwerk-Interface-Einheiten
● die Netzwerk-Interface-Einheiten NIU (Network Interface Unit) für den Anschluß der Teilnehmerstationen an das Übertragungssystem
● die Teilnehmerstationen (Teilnehmerrechner) mit
– Kommunikationssoftware
– Basis-/Anwendungssoftware für Standardanwendungsdienste (Filetransfer, Jobverarbeitung, Dokumentenübertragung, elektronischer Post usw.) und
– spezialisierter Software z. B. für die Prozeß- und Fertigungssteuerung
● Koppeleinrichtungen (Gateways) zur Verbindung lokaler Netze untereinander oder mit globalen Rechnernetzen.

Als topologische Grundstrukturen haben sich Bus- und Ringstrukturen bewährt.
Ein wichtiges Charakteristikum lokaler Netze ist die Methode des Zugriffs zum Übertragungsmedium. Die international am weitesten verbreiteten Methoden sind CSMA/CD und Token.
CSMA/CD-LAN sind charakterisiert durch stochastischen Zugriff zum Übertragungsmedium. Sie sind relativ einfach zu realisieren. Etwa 60 Prozent aller bekannten LAN-Einsatzfälle mit den Anwendungen CAD, Büroautomatisierung, lokaler Rechnerverbund und Kopplung von Automatisierungseinrichtungen mit Maschinensteuerungen arbeiten nach diesem Prinzip. Bei hoher Belastung kann der Zugriff der Teilnehmerstationen zum Übertragungsmedium in ausreichend kurzer Zeit nicht mehr garantiert werden. Das führt zur Einschränkung in der Anwendbarkeit z. B. für bestimmte Steuerungsaufgaben.
Token-LAN sind gekennzeichnet durch deterministischen Zugriff zum Übertragungsmedium. Ihre Realisierung ist aufwendiger als die der CSMA/CD-LAN. Wegen der garantierten maximalen Verzögerungszeiten beim Zugriff der Teilnehmerstationen zum Übertragungsmedium werden sie vor allem bei der Prozeßautomatisierung und in Steuerungssystemen eingesetzt. Dieses LAN-Prinzip wird in etwa 30 Prozent der gegenwärtigen Einsatzfälle angewendet.
Eine ausführliche Darstellung der LAN-Problematik erfolgt in /4/.

Schichten-Architektur

Seit 1977 wird international an der Standardisierung der rechnergestützten Kommunikation gearbeitet. Ein Ergebnis von grundsätzlicher Bedeutung liegt seit 1983 als *Standard Informationsverarbeitungssysteme – Verkopplung offener Systeme – Basisreferenzmodell* /5/ der Internationalen Standardisierungsorganisation (ISO) vor.
Das darin spezifizierte OSI-Referenzmodell (Bild 2) ist ein Schichtenmodell für Kommunikationsvorgänge, das eine Orthogonalität der den Schichten zugeordneten Übertragungsfunktionen gewährleistet. Hauptaspekte dieses Schichtenmodells sind:

Fortsetzung von Seite 73

/8/ Molčanov, I. N.; Rjabcev, V. E.: Ob organizacii vyčislenij pri rešenii bol'šich sistem linejnych algebraičeskich uravnenij metodom otraženij. Kibernetika (1983) 2, S. 24–28 (Russ.)
/9/ Molčanov, I. N.; Chimič, A. N.: Nekotorye algoritmy rešenija simmetričnoj problemy sobstvennych značenij na mnogoprocessornoi elektronnoj vyčislitel'noi mašine (MEVM). Kibernetika (1983) 6, S. 36–38. (Russ.)
/10/ Molčanov, I. N.; Jakovlev, M. F.; Geec, E. T.: Nekotorye metody integrirovanija obyknovennych differencial'nych uravnenij na mnogoprocessornoi EVM. Kibernetika (1983) 5, S. 10–13 u. S. 21 (Russ.)
/11/ Viegas de Vasconcelos, A.: Multiprozessoren und ihre Anwendung zur Analyse elektrischer Netze. Elektronische Rechenanlagen 26 (1984) 3, S. 145–151

KONTAKT
Technische Hochschule Magdeburg, Sektion Rechentechnik und Datenverarbeitung, Boleslaw-Bierut-Platz 5, Magdeburg, 3010; Tel. 59 28 76

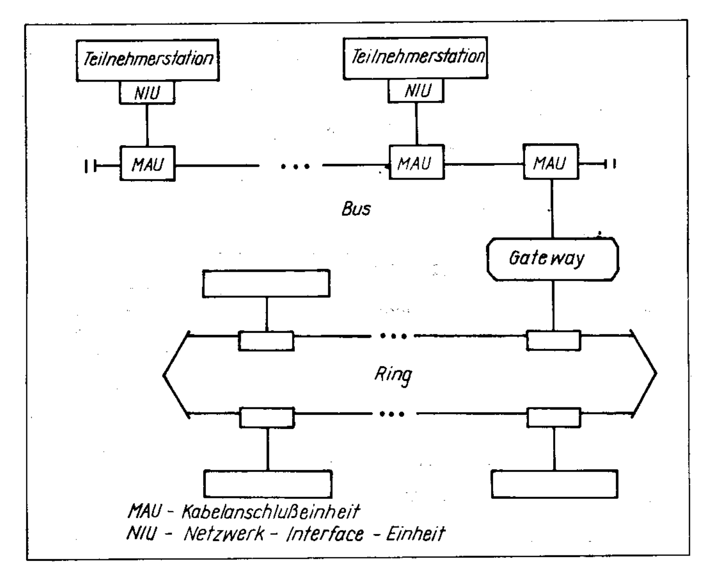



Bild 1 LAN-Hauptkomponenten

OSI-Schicht	Standards für LAN	Standards für WAN
Anwendungsschicht	ISO 8649, ISO 8650, ISO 8571, ...	
Darstellungsschicht	ISO 8822, ISO 8823	
Sitzungsschicht	ISO 8326, ISO 8327	
Transportschicht	ISO 8072, ISO 8073	
Netzschicht	ISO 8348, ISO 8473, ISO 8648, ... ISO 8802/1	X. 25/3
Verbindungsschicht	ISO 8802/2	X. 25/2 (LAPB)
Physikalische Schicht	ISO 8802/3, ISO 8802/4, ISO 8802/5, ISO 8802/6	X. 25/1 (X. 21, V. 35, ...)

Bild 3 Internationale Standards entsprechend der einheitlichen Architektur für LAN und WAN

- Inhaltlich zusammenhängende Funktionen sind in einer Schicht zusammengefaßt. Die Schnittstellen zwischen den Schichten sind festgelegt und unabhängig von der konkreten Implementierung (bzw. vom physikalischen Übertragungsmedium). Damit ist es z. B. möglich, eine Softwareschicht durch eine neue Implementierung dieser Schicht auszutauschen oder ein Koaxialkabel durch einen Lichtwellenleiter zu ersetzen, ohne daß die anderen Schichten geändert werden müssen.
- Die Kommunikationsbeziehungen zwischen Komponenten der gleichen Schicht (in den Endgeräten) werden durch Protokolle festgelegt. Der Informationsaustausch zwischen benachbarten Schichten eines Systems wird über Dienstschnittstellen geregelt.
- Jede Schicht nimmt für die Realisierung des von ihr bereitzustellenden Kommunikationsdienstes die Dienste der darunterliegenden Schicht (im Fall der Schicht 1 des physikalischen Mediums) in Anspruch.

In einem *offenen* Rechnersystem im Sinne des Modells werden OSI-Standards zur Kommunikation angewendet und dadurch die Zusammenarbeit mit anderen Rechnersystemen, in denen diese ebenfalls eingehalten werden, ermöglicht.
Ausführlicher kann dazu z. B. in /6/ nachgelesen werden. Nachdem das OSI-Referenzmodell ursprünglich für globale Rechnernetze (Wide Area Network – WAN) erarbeitet wurde, ist es jetzt international auch für LAN als Grundarchitektur akzeptiert. Die Integration LAN-spezifischer Standards in das Modell erfolgt gegenwärtig mittels Adaption der IEEE P802-LAN-Standards als ISO-Standards /7/ (Bild 3).

Vor dem Hintergrund einer Vielzahl existierender LAN wird international an standardgerechten Lösungen gearbeitet, die sich durch eine hocheffektive Implementierung der Kommunikationsfunktionen bei geringer Belastung der anzuschließenden Teilnehmerstationen auszeichen sollen.
Zur Zeit werden in der Regel die Schichten 1 und 2 über intelligente Controller als Steckeinheiten mit eigenem Prozessor und Ergänzungsbaugruppen bzw. mit Spezial-VLSI-Schaltkreisen (LANC, SIA) und die darüberliegenden Schichten über Softwarekomponenten in den Teilnehmerstationen realisiert. Die Tendenz geht zur durchgängigen Verwendung der VLSI-Schaltkreise für die Schichten 1 und 2 und zum Einsatz von LAN-Kopplern, die als spezielle Hard/Softwarekomponenten die Funktionen der Schichten 1 bis 4 wahrnehmen. Diese LAN-Koppler werden als Rechnereinschub oder als eigenständige Geräte, die zusätzlich Konzentratorfunktionen wahrnehmen können, bereitgestellt.
Die einheitliche Architektur für lokale und globale Netze ermöglicht deren Verkopplung mit gegenüber bisherigen Lösungen deutlich reduziertem Aufwand. Die Schnittstelle zwischen einheitlichen und netzspezifischen Standards für LAN und WAN liegt in der Schicht des OSI-Referenzmodells. Demzufolge brauchen in den zur Verkopplung erforderlichen Anpassungseinrichtungen (Gateways) nur die unteren drei Schichten aufeinander abgebildet zu werden. Darüber hinaus ermöglicht die Einheitlichkeit der oberen vier Schichten, die entsprechende Software nur einmal zu implementieren und in LAN und WAN einzusetzen.

offenes System | Protokolle | offenes System

Schicht	offenes System	Protokolle	offenes System
7	Anwendungsschicht	←→	
6	Darstellungsschicht	←→	
5	Sitzungsschicht	←→	
4	Transportschicht	←→	
3	Netzschicht	←→	
2	Verbindungsschicht	←→	
1	Physikalische Schicht	←→	

Physikalisches Übertragungsmedium

Bild 2 OSI-Referenzmodell zur Verkopplung offener Systeme

Prof. Heymer studierte Mathematik an der Moskauer Staatlichen Lomonossow-Universität und promovierte zum Dr. rer. nat. und Dr. sc. techn. an der Akademie der Wissenschaften der DDR. Nach Arbeiten auf dem Gebiet der Betriebssystementwicklung sind seit 1975 die Rechnernetze sein Spezialgebiet. Seine speziellen Interessen umfassen Rechnernetzarchitektur und -protokolle, kommunikationsorientierte Implementierungsumgebungen, Dienst- und Protokollspezifikationsmethoden sowie Konformitätsbewertung.

Schlußbemerkungen

Der Bedarf an LAN-Lösungen nimmt in der DDR ständig zu und wird mit der weiteren breiten Anwendung von CAD- und CAM-Systemen in den nächsten Jahren sprunghaft ansteigen.
Eine effektive Verkopplung der unterschiedlichen für diese Aufgaben eingesetzten Rechnersysteme (Personal-Computer, Kleinrechner, ESER-EDVA) aus der Produktion der RGW-Staaten erfordert eine durchgängige

Standardisierung auf der Basis des OSI-Referenzmodells. Dazu werden im Bereich des Hochschulwesens und der AdW der DDR sowie im Kombinat Robotron ebenso wie in den anderen sozialistischen Ländern Vorlauf- und Entwicklungsarbeiten mit dem Ziel durchgeführt, die produzierte Rechentechnik LAN-fähig zu gestalten.

Literatur

/1/ Boitschenkov, E. W. u. a.: Lokale Rechnernetze. Radio i Svjaz, Moskau, 1985

/2/ Metcalfe, R. M.; Boggs, D. R.: Ethernet: Distributed packet switching for local computer networks. Comm. ACM 19 (1976) 7

/3/ Yakubaitis, E. A.: Lokale Rechnernetze. Sinatne, Riga, 1985

/4/ Löffler, H.: Lokale Netze. Akademie-Verlag Berlin (Buchmanuskript 1986). erscheint 1987

/5/ Information Processing Systems – Open Systems Interconnection – Basic Reference Model, ISO/IS7498

/6/ Heymer, V.: Verkopplung von Rechnersystemen zu Rechnernetzen. ZfR-Informationen, ZfR-84. 13, 1984

/7/ Data communication – Local Area Networks (LAN) – Standards ISO DIS 8802/1 . . . DIS 8802/6

Fortsetzung von Seite 71

/8/ Claßen, L.: CP/M – ein Floppy-Disk-orientiertes Betriebssystem für Arbeitsplatzcomputer mit 8-Bit-Mikroprozessoren. msr, Berlin 27 (1984) 11, S. 500–504

/9/ Meichsner, H.: Drei Betriebssysteme – SIOS, UDOS, SCP – laufen auf dem Bürocomputer robotron A 5120/A 5130. Neue Technik im Büro, Berlin 28 (1984) 3, S. 93–95

✉ KONTAKT ☎

Kombinat VEB Elektro-Apparate-Werke Berlin-Treptow, Zentrum für Forschung und Technologie, Abt. Basissoftware, Hoffmannstr. 15–26, Berlin, 1193, Tel. 4 38 87 30; Anfragen zum Vertrieb an Direktorat Absatz, Koll. Peschke.

AD31 – ein monolithisch integrierter A/D-Umsetzer

Jürgen Schmidt
Technische Hochschule Ilmenau

Vorbemerkungen

Für die Entwicklung von monolithisch integrierten MOS-A/D-Umsetzern besitzt das Verfahren der sukzessiven Approximation eine große Bedeutung, da hierbei ein Optimum hinsichtlich Wandlungszeit (kleiner 100 µs), Auflösung (8 . . . 12 Bit) und der Verwendung weniger kritischer Analogbaugruppen erreichbar ist. Weiterhin stellt der hohe Anteil an Steuerlogik kein Problem für einen MOS-freundlichen Entwurf dar. Bild 1 zeigt das Blockschaltbild eines solchen A/D-Umsetzers. Bei dem sukzessiven Approximationsverfahren wird das Bitwort durch schrittweise Annäherung der Ausgangsspannung eines D/A-Umsetzers an die Eingangsspannung ermittelt. Eine genaue Beschreibung dieses Verfahrens ist in /1/ zu finden. Für den internen D/A-Umsetzer bietet sich in der MOS-Technologie ein Kapazitäts-Schalter-Netzwerk an; dafür sprechen nachfolgend genannte Vorteile (siehe dazu auch /2/):

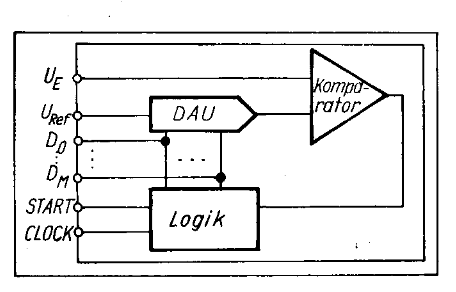


Bild 1 A/D-Umsetzer nach dem sukzessiven Approximationsverfahren

- Kompatibilität zur MOS- bzw. CMOS-Technologie
- 8 . . . 10 Bit Auflösung ohne zusätzlichen Abgleich
- Nutzung der Gesamtkapazität für eine interne sample-and-hold-Stufe
- geringer Einfluß der realen Eigenschaften der Analogschalter
- keine statische Belastung der Referenzspannung
- vernachlässigbarer Temperatureinfluß.

Des weiteren ist, auch aus der Literatur, bekannt, daß die Realisierung hochgenauer Kapazitätsverhältnisse relativ einfach möglich ist, wobei die dabei erreichbaren Werte denen präziser Dünnfilmwiderstandsnetzwerke sehr nahe kommen /2, 3, 4/.

Im Rahmen einer gemeinsamen Forschungsaufgabe der Technischen Hochschule Ilmenau, Sektionen Informationstechnik und theoretische Elektrotechnik sowie Physik und Technologie elektrotechnischer Bauelemente, und des Technikums des VEB Mikroelektronik „Karl Marx" Erfurt entstand der im folgenden beschriebene A/D-Umsetzer.

Funktionsbeschreibung

Der IS AD31 ist ein Testschaltkreis, der einen vollständigen A/D-Umsetzer nach dem sukzessiven Approximationsverfahren beinhaltet und in der n-Kanal-Silizium-Gate-Technologie präpariert wurde. Auf einer Chipfläche von 2,5 µm mal 3,5 µm wurden dazu 1150 Transistoren und 260 diskrete Kapazitäten realisiert.

Als D/A-Umsetzer wurde das in Bild 2 dargestellte Netzwerk mit binärer Wichtung der Kapazitäten eingesetzt, in welchem das Verfahren der kapazitiven Spannungsteilung angewendet wird.

Um die maximal zulässigen Abweichungen der Kapazitätsverhältnisse von kleiner 0,39 % für einen 8-Bit-Umsetzer /5/ sicher einhalten zu können, wurden die Kapazitäten durch parallel geschaltete Einheitskapazitäten vom Wert C_0 realisiert, wobei auf die Symmetrie des Netzwerkes und auf gleiche geometrische Randbedingungen aller Einheitskapazitäten besonderer Wert gelegt wurde.

Der D/A-Umsetzer und der Komparator sind die beiden genauigkeitsbestimmenden Baugruppen des A/D-Umsetzers. Dabei muß der Komparator den beiden Forderungen nach großer Verstärkung und schnellem Schalten gerecht werden, weshalb ein 4stufiger kapazitiv gekoppelter Chopperverstärker mit nachgeschaltetem D-Latch eingesetzt wurde. Die Logik ist, um bezüglich der benötigten Transistoren, unkritischer Taktung bei Taktfrequenzen bis 10 MHz, Leistungsaufnahme, Technologie- und Temperaturabhängigkeit ein Optimum zu erreichen, in semistatischer Schaltungstechnik ausgeführt. Sie umfaßt die Baugruppen 8-Bit-Approximationsregister, 10-Bit-Steuerschieberegister, Ausgaberegister mit tristate-Treibern und Takt- und Ablaufsteuerung.

Mit Hilfe des Blockschaltbildes (Bild 3) wird im folgenden die Funktion des A/D-Umsetzers beschrieben. Dazu wird der zeitliche Ablauf in die 3 Zeitphasen Abtast-, Halte- und Approximationsphase unterteilt /6/.

Abtastphase

Der Schalter S_A ist geschlossen, und alle Kapazitäten sind über die Schalter S_0 bis S_8 zwischen Referenzspannung U_{Ref} und Eingangsspannung U_E geschaltet. Damit folgt die Ladung über allen Kapazitäten gemäß

$$Q(t) = (U_{Ref} - U_E(t)) * C_{ges} \qquad (1)$$

der Eingangsspannung $U_E(t)$.

Haltephase

Die Haltephase wird mit dem Öffnen des Schalters S_A eingeleitet, womit auch

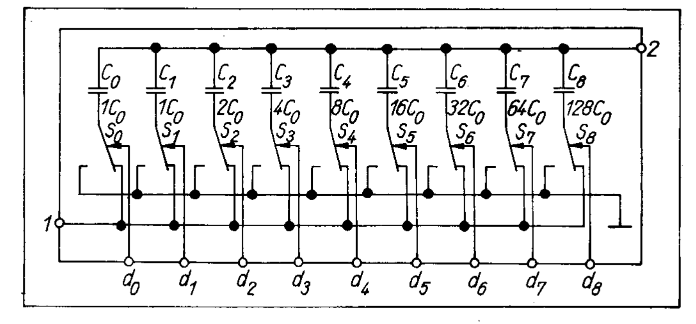


Bild 2 8-Bit-D/A-Umsetzer mit binärgewichteten Kapazitäten

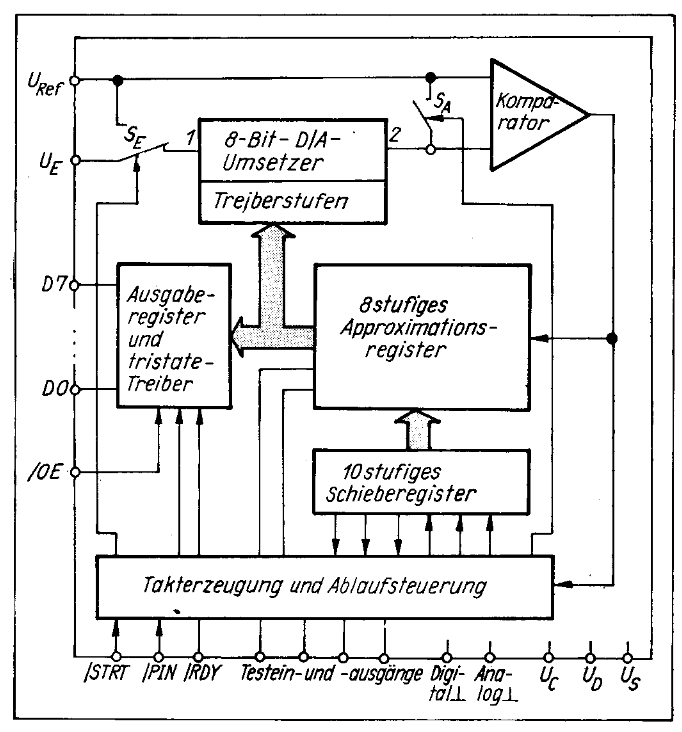


Bild 3 Blockschaltbild des Testschaltkreises AD31

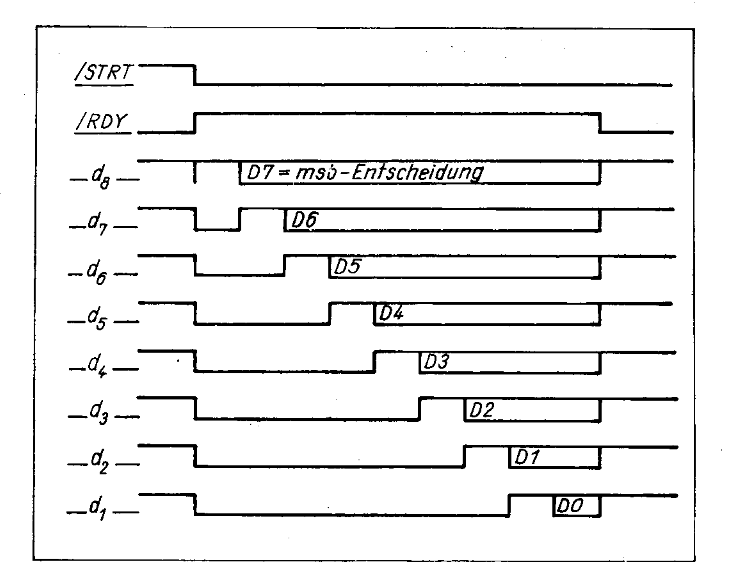


Bild 4 Zeitdiagramm des Testschaltkreises AD31

der Umsetzvorgang selbst gestartet wird. Dabei ist die auf allen Kapazitäten gespeicherte Ladung der Eingangsspannung zum Zeitpunkt des Öffnens des Schalters S_A ($U_E(t_0)$) äquivalent. Dieser Sachverhalt entspricht der Wirkungsweise einer sample-and-hold-Schaltung.

Approximationsphase

Die Approximationsphase folgt unmittelbar auf die Auslösung der Haltephase. Von der Steuerlogik werden die Schalter S_0 bis S_8 an den Kapazitäten auf Masse geschaltet, weiterhin wird S_E gegen die Referenzspannung U_{Ref} geschaltet. S_8, dieser Schalter repräsentiert das MSB D_7, wird nun gegen die Referenzspannung U_{Ref} geschaltet, so daß mittels des entstandenen kapazitiven Spannungsteilers dem vorhandenen Potential ($U_{Ref} - U_E(t_0)$) das Potential der halben Referenzspannung am Anschlußpunkt 2 gemäß dem Superpositionsprinzip überlagert wird. Nachdem das System eingeschwungen ist, vergleicht der Komparator dieses Potential mit der Referenzspannung. Ist es größer U_{Ref}, so wird d_8 zurückgesetzt (damit wird $D_7 = L$), ansonsten bleibt d_8 gesetzt (gegen U_{Ref} geschaltet, damit ist $D_7 = H$). Gleichzeitig wird d_7 gegen U_{Ref} geschaltet (gesetzt), damit wird dem vorhandenen Potential am Anschlußpunkt 2 das von $1/4(U_{Ref})$ überlagert. Danach wird der Vergleichsprozeß durchgeführt und damit der Wert von D_6 ermittelt. Dies wiederholt sich, bis alle Bitwertigkeiten ermittelt sind. Damit nähert sich das Potential am Anschlußpunkt 2 schrittweise dem der Referenzspannung U_{Ref}, so daß die verbleibende Differenzspannung U_D am Komparator gleich dem Quantisierungsfehler U_q wird. Für

$$U_D = \qquad (2)$$

$$(U_{Ref} - U_E(t_0)) + U_{Ref} * \sum_{i=1}^{n} d_i k_i - U_{Ref}$$

mit $U_D = U_q$, $|U_q| \leqq | \pm 1/2 LSB$

folgt somit für die Eingangsspannung U_E

$$U_E = U_{Ref} * \sum_{i=1}^{n} d_i k_i - U_q \,, \qquad (3)$$

wobei k_i der Teilungsfaktor der jeweiligen Bitstelle d_i ist ($k_8 = 1/2$, $k_7 = 1/4$, $k_6 = 1/8, \ldots, k_1 = 1/256$). Die Kapazität C_0 mit dem Schalter S_0 dient dabei nicht dem prinzipiellen Verfahren, sondern der Minimierung des Verstärkungsfehlers.

Ansteuerung des AD31

Die Ansteuerung des Testschaltkreises ist relativ unproblematisch, das zeitliche Verhalten ist den Bildern 4, 5 und 6 zu entnehmen. Der ADU kann im Start-Stop-Modus oder kontinuierlich betrieben werden, die Auswahl zwischen beiden Betriebsarten erfolgt über die Belegung des Start-Einganges (/STRT). Wird dieser Eingang ständig mit L beschaltet, so setzt der ADU kontinuierlich um, wobei der Ready-Ausgang (/RDY) den Beginn einer Umsetzung (Abtastzeitpunkt t_0) und die Übergabe des A/D-gewandelten Datenwortes an das Ausgaberegister signalisiert. Wird /STRT mit einem L-Impuls belegt (dieser Impuls muß kürzer als 44 Taktperioden des extern anzulegenden Taktes CPIN sein), so erfolgt nur eine A/D-Umsetzung, nach deren Abschluß der ADU bis zur nächsten HL-Flanke an /STRT in der Abtastphase verweilt (Start-Stop-Modus). Der /RDY-Ausgang hat hierbei die gleiche Funktion wie beim kontinuierlichen Betrieb des ADU.

Jürgen Schmidt (31) studierte von 1977 bis 1982 Informationstechnik an der TH Ilmenau und diplomierte mit einer Arbeit über spezielle Untersuchungen an geschalteten Kapazitätsnetzwerken. Danach war er 4 Jahre als wissenschaftlicher Assistent an der TH Ilmenau tätig. Hauptgegenstand der wissenschaftlichen Arbeit war der Entwurf eines monolithisch integrierten Analog-Digital-Umsetzers in NMOS-Technik. Seit 1986 arbeitet Jürgen Schmidt am Entwurf kundenspezifischer Schaltkreise im VEB Kombinat Carl Zeiss JENA.

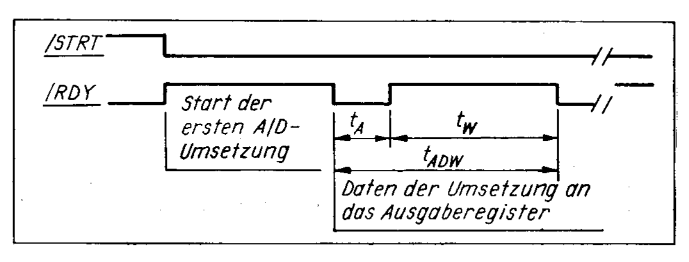


Bild 5 Signalverhalten im kontinuierlichen Modus

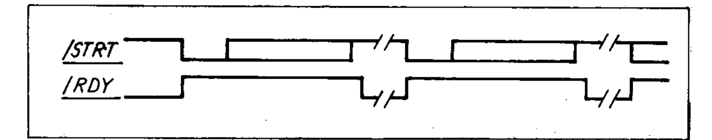


Bild 6 Signalverhalten im Start-Stop-Modus des ADU

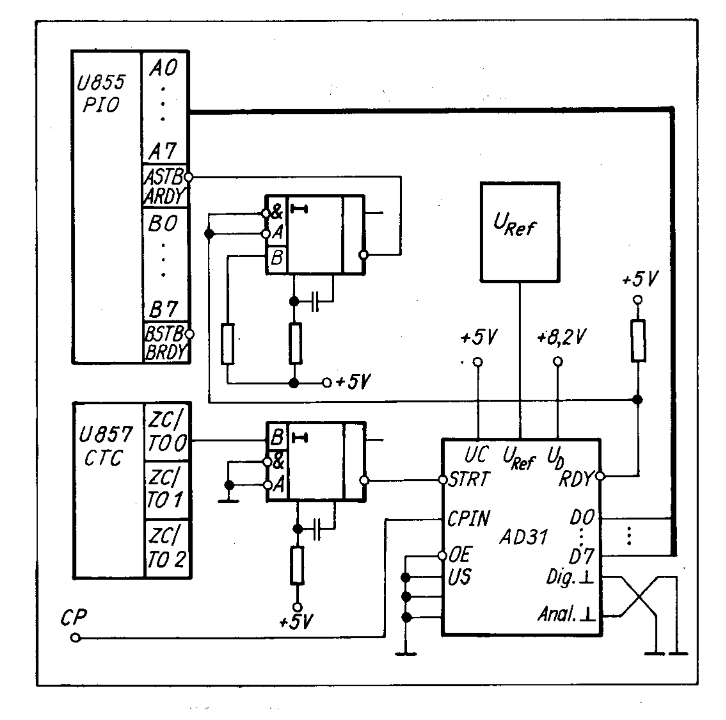


Bild 8 Applikationsbeispiel zum Testschaltkreis AD31

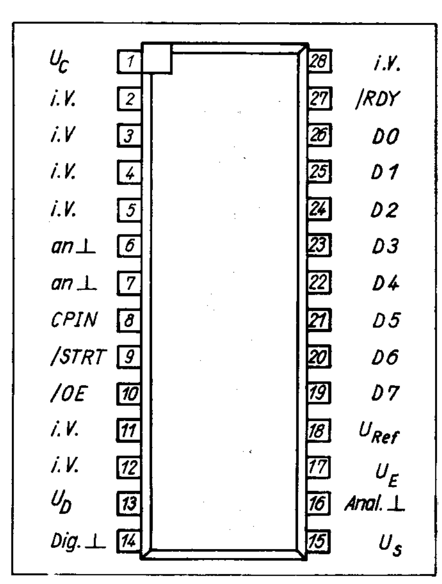


Bild 7 Anschlußbelegung des Testschaltkreises AD31

Der /OE-Eingang steuert mit /OE = H die tristate-Ausgabetreiber in den hochohmigen Zustand, damit ist der Anschluß des ADU an einen beliebigen 8-Bit-Bus möglich.
Bei einer Taktfrequenz von 4,4 MHz an CPIN gilt für den kontinuierlichen Betriebsfall:

- *Umsetzzeit* $t_W = 8{,}3\,\mu s$
- *Abtastzeit* $t_A = 1{,}7\,\mu s$
- *Abtastrate* $1/t_{ADW} = 100\,kHz$

bzw. allgemein:

$$t_A = 7{,}5/CPIN,\ t_W = 36{,}5/CPIN;\quad t_{ADW} = t_A + t_W = 44/CPIN. \tag{4}$$

Damit läßt sich der Testschaltkreis AD31 für die A/D-Umsetzung von Signalfrequenzen im NF-Bereich bis 50 kHz einsetzen, wobei die intern realisierte sample-and-hold-Stufe den äußeren schaltungstechnischen Aufwand vermindert.

Technische Angaben

Die im Kasten aufgeführten vorläufigen technischen Daten resultieren aus Messungen an den Mustern einer Testpräparation; die zeitabhängigen Parameter gelten für eine Taktfrequenz von CPIN = 4,4 MHz. Die relativ großen Linearitätsfehler treten an charakteristischen Stellen der Umsetzerkennlinie auf und sind im wesentlichen auf eine ungünstige Kopplung zwischen internem D/A-Umsetzer und Komparator zurückzuführen. Eine Korrektur kann nur durch eine Überarbeitung der Schaltung und damit des Layoutentwurfs realisiert werden.
(Die Anschlußbelegung wird in Bild 7 gezeigt.)

Vorläufige technische Daten des ADU AD31

• *Auflösung*	8 Bit
• *Umsetzzeit*	10 µs (mit Abtastzeit)
• *Umsetzrate*	maximal 100 kHz
• *Erfassungszeit der internen sample-and-hold-Stufe*	1,7 µs (für kontinuierlichen Modus)
• *Eingangsspannungsbereich*	0 . . . 5,12 V
• *Referenzspannung*	*5,12 V (extern)*
• *Übertragungscharakteristik*	linear
• *Ausgangscode*	Binärcode
• *Datenausgänge*	TTL-kompatibel, fan-out=2, tristate
• *RDY-Ausgang*	TTL-kompatibel, fan-out=2, open-drain
• *Steuereingänge*	TTL-kompatibel
• *Takteingang CPIN*	TTL- oder 5-V-CMOS-Einphasentakt, k = 0,5, max. Frequenz = 4,4 MHz
• *Betriebsspannungen*	UC = 5 V, U_D = 8 . . . 12 V, U_S = 0 V
• *Leistungsaufnahme*	200 . . . 300 mW
• *Innenwiderstand der analogen Quellen (U_E, U_{Ref})*	kleiner 2 kOhm
• *Linearitätsfehler*	kleiner 5 LSB
• *Offsetfehler*	kleiner 2 LSB

Applikationsbeispiel

In dem im Bild 8 gezeigten Applikationsbeispiel wird der AD31 an einem 8-Bit-Mikrorechner-Interface betrieben. Dazu belegen die 8 Datenausgänge des ADU den Kanal A einer PIO. Da der ADU im Start-Stop-Modus betrieben wird, wird die von einem CTC-Baustein generierte Abtastfrequenz mit Hilfe eines Monoflops zu Start-Impulsen für den ADU aufbereitet. Das RDY-Signal wird mit Hilfe eines Monoflops zu Impulsen geformt, die über /ASTB der PIO einen Interrupt anfordern können. Wenn erforderlich, können mit Hilfe des /OE-Einganges mehrere ADU parallel geschaltet werden, womit mehrere analoge Quellen zeitmultiplex abgearbeitet werden können.

Literatur

/1/ Bobe, W.: Schneller Analog-Digital-Umsetzer mit TTL-Schaltkreisen. Radio, Ferns., Elektron. Berlin 27 (1978) 1, S. 655, 656

/2/ Mc Creary, J. L.; Gray, P. R.: All-MOS Charge Redistribution Analog-to-Digital Conversion Techniques-Part 1. IEEE Journal of Solid-State Circuits, Dec. 1975, S. 371–379

/3/ Dahms, J.: MOS-Kondensatoren über hochdotiertem Substrat in integrierbaren ADU. Dissertation an der Universität Dortmund, Abteilung Elektrotechnik, Dortmund 1978

/4/ Keßler, H.; Roßgotterer, R.: Ein 8-Bit-Analog-Digital-Wandler in MOS-Silizium-Gate-Technologie nach dem Ladungsteilungsverfahren. Siemens Forsch. – und Entwickl. – Bericht, Band 8 (1979) Nr. 5, S. 261–263

/5/ Schmidt, J.: Spezielle Untersuchungen an geschalteten Kapazitätsnetzwerken. Diplomarbeit an der TH Ilmenau, Sektion INTET, Ilmenau 1982

/6/ Schmidt, J.; Teich, W.; Stöhr, H.: Testschaltkreis AD31 – Ein monolithisch integrierter 8-bit-A/D-Wandler. TU Dresden, 19. Fachkolloquium Informationstechnik, Tagungsbericht Teil 2, Dresden 1986

Programmierung in C

(Teil III)

Dr. Thomas Horn
Informatik-Zentrum des Hochschulwesens an der Technischen Universität Dresden

Nachdem in den ersten beiden Teilen dieser Artikelreihe die Lexik, Datentypen, Speicherklassen, Operatoren, Ausdrücke und Steuerstrukturen erläutert wurden, werden in diesem Teil das Zeigerkonzept und die Arbeit mit Feldern beschrieben. Insbesondere das Zeigerkonzept ist für die Systemprogrammierung von Bedeutung, da erst die Arbeit mit Adressen eine effektive Lösung vieler Probleme gestattet und auch gegenüber der Arbeit mit Feldindizes Vorteile hat. Hierin drücken sich vor allem die Vorteile von C als effektive Systemprogrammiersprache aus.
Die Felder dienen zur Zusammenfassung von Datenobjekten gleichen Typs. Im weiteren werden dann die Strukturen und Vereinigungen zur Zusammenfassung von Datenobjekten verschiedener Typen behandelt.

5. Zeiger

Die Programmiersprache C enthält ein ausgezeichnetes Zeigerkonzept. Unter einem Zeiger (*pointer*) versteht man ein Datenobjekt, das eine Adresse enthält, über die auf beliebige andere Datenobjekte zugegriffen werden kann. Intern werden Zeiger wie **unsigned**-Variablen dargestellt. Sie dürfen aber nicht mit **unsigned**-Variablen verwechselt werden, da Zeiger zusätzlich über ein Längenattribut verfügen, das die Länge des Datenobjekts in Byte spezifiziert, auf das der Zeiger zeigt. Das Längenattribut ist insbesondere für Adreßrechnungen wichtig. Das Zeigerkonzept wird vor allem in der Systemprogrammierung genutzt, weil viele Probleme nur über die Verwendung von Adressen effektiv gelöst werden können. Auch in anderen Anwendungen bringt die Arbeit mit Adressen Effektivitätsvorteile gegenüber der Arbeit mit Feldindizes.
Das Zeigerkonzept birgt jedoch generell die Gefahr in sich, daß durch die Arbeit mit Adressen die Portabilität der Programme eingeschränkt wird. In C ist das Zeigerkonzept so implementiert, daß bei einer sorgfältigen Programmierung eine Verletzung des Portabilitätsprinzips ausgeschlossen werden kann.

5.1. Zeigervariablen

Wie alle Variablen müssen Zeigervariablen zu Beginn eines Blockes vereinbart werden, wobei der Indirektoperator (*) vor dem Identifikator die Variable als Zeigervariable und der Typbezeichner ihren Typ spezifizieren.

Beispiel:
```
int * pi;
char * pc;
float * pf;
```

Die Variable **pi** ist ein Zeiger auf eine Integervariable, **pc** ein Zeiger auf eine Zeichenvariable und **pf** ein Zeiger auf eine Gleitkommazahl einfacher Genauigkeit.
Adressen werden von einfachen Variablen über den Adreßoperator (**&**) gebildet.

Beispiele:
```
pi=&A;
pc=&buffer [0];
pf=&sum;
```

Der Zugriff auf die Speicherobjekte erfolgt dann über einen Zeiger durch Anwendung des Indirektoperators (*) auf die Zeigervariable.

Beispiel:
```
pi=&A;
B=*pi;
```

Durch die letzte Anweisung wird der Variablen **B** der gleiche Wert wie bei der Anweisung **B=A** zugewiesen. Da **pi** als ein Integerzeiger vereinbart ist, besagt die Kombination ***pi**, daß das Speicherobjekt ein Integerwert ist.

Zeigervariablen können auch in Ausdrücken auftreten.

Beispiele:
```
B=*pi+1;
```

Durch diese Anweisung wird das Speicherobjekt ***pi** gelesen und anschließend um 1 erhöht. Das Ergebnis entspricht somit **B=A+1**

```
B=*(pi+1);
```

Durch diese Anweisung wird der Zeiger **pi** um 1 erhöht. Da es sich um einen Integerzeiger handelt, wird der Zeiger auf das nächste Integerobjekt weiter gestellt (Erhöhung der Adresse um 2 Byte), welches dann gelesen wird.
Diese Operation wird oft bei Feldern angewendet, um das nächste Feldelement zu adressieren.
Zeiger können auch auf der linken Seite von Zuweisungen verwendet werden.

Beispiel:
```
*pi *=5;
```

Das Speicherobjekt vom Typ **int**, das durch **pi** adressiert wird, wird mit fünf multipliziert und auf den selben Speicherplatz abgelegt.

Hinweis:

Das Zeigerkonzept kann nicht auf Variablen der Speicherklasse **register** angewendet werden, da von Registern keine Adressen gebildet werden können.

5.2. Adreßrechnungen

Unter Adreßrechnungen werden Operationen mit den Adressen verstanden, die in den Zeigervariablen gespeichert sind. Bei den Adreßrechnungen ist zu beachten, daß alle arithmetischen Operationen unter Beachtung der Längenattribute der Zeiger ausgeführt werden.

Die Adreßarithmetik läßt folgende Operationen mit Zeigern zu:

- Addition einer ganzen Zahl zum Zeigerwert
  Der Zeiger wird um eine ganze Zahl von Datenobjekten vorwärts gerückt, z. B. **pf += 5;** oder **pi++;**
- Subtraktion einer ganzen Zahl vom Zeigerwert
  Der Zeiger wird um eine ganze Zahl von Datenobjekten rückwärts gesetzt, z. B. **pc -= 3;** oder **pf--;**
- Subtraktion von zwei Zeigern
  Die Differenz von zwei Zeigern ist die Anzahl von Datenobjekten, die sich zwischen zwei Zeigern befinden, z. B. **i = pf2 - pf1;**
- Vergleich von zwei Zeigern
  Der Vergleich von zwei Zeigern ergibt, ob zwei Zeiger identisch sind oder nicht bzw. ob ein Zeiger größer oder kleiner als der andere ist, z. B. **if (pf1 < pf2)...;**

Hinweise:

– Zwei Zeiger können nicht addiert werden.
– Zeiger können nicht multipliziert, dividiert, verschoben oder mit anderen logischen Operationen verändert werden.
– Es können keine Additionen oder Subtraktionen mit **float-** oder **double-**Werten ausgeführt werden.

Beispiel:
Es seien die Zeigervariablen **pi**, **pc** und **pf** definiert (siehe 5.1.).

```
pi++;
pc++;
pf++;
```

Der Inkrementoperator **++** führt eine Erhöhung der Zeiger auf das nächste Speicherobjekt durch. Bei **pi** wird somit die Adresse um zwei Bytes, bei **pc** um ein Byte und bei **pf** um vier Bytes erhöht. (Die Länge eines Speicherobjektes kann mit dem **sizeof**-Operator bestimmt werden.)
Zur Absicherung der richtigen Ausführung von Adreßoperationen kann von der expliziten Zeigerumwandlung Gebrauch gemacht werden.

Beispiel:

```
pc=(char*)pi;
```

Der Zeiger vom Typ **int** wird in einen Zeiger vom Typ **char** umgewandelt.

Analog können auch Integerwerte in Zeiger und umgekehrt umgewandelt werden. Auf verschiedenen Rechenanlagen können hierbei Wortgrenzenfehler entstehen. Deshalb sollten die Zeigerumwandlungen mit größter Vorsicht verwendet werden.

5.3. Initialisierung von Zeigern

Im allgemeinen kann ein Zeiger wie jede andere Variable initialisiert werden. Sinnvolle Werte sind allerdings nur die Adressen bereits vorher definierter Objekte. In C ist sichergestellt, daß nicht initialisierte Zeiger bzw. Zeiger, die nicht korrekt auf ein Speicherobjekt verweisen, den Wert 0 erhalten. Die Adresse 0 ist niemals die Adresse eines gültigen Speicherobjekts. Man kann deshalb auch den Wert 0 benutzen, um eine Fehlersituation anzuzeigen. Damit kann über den Zeiger auf kein Speicherobjekt mehr zugegriffen werden. Um sichtbar zu machen, daß der Wert 0 eine besondere Bedeutung hat, wird das Symbol **NULL** verwendet, das im File **STDIO.H** (siehe 10.1.) definiert ist.

Beispiel:

```
char text [2000];
static char *t=&text [0];
```

Durch die Initialisierung wird der Zeiger **t** auf den Anfang des Textpuffers **text** gesetzt.

```
if (t==&text [1999]) t=NULL;
```

Wenn das Ende des Textpuffers erreicht ist, wird **t** auf **NULL** gesetzt, was an späterer Stelle im Programm wieder getestet werden kann:

```
if (t==NULL) goto ENDE;
```

6. Felder

Felder werden zur Zusammenfassung von Datenobjekten des gleichen Typs zu einer Gesamtheit verwendet. Die Datenobjekte werden dann unter dem gemeinsamen Feldnamen geführt. Auf die einzelnen Datenobjekte, auch Feldelemente genannt, kann über Indizes zugegriffen werden. Werden die Feldelemente linear angeordnet, so können sie über einen Index adressiert werden. Man spricht von einem eindimensionalen Feld. Bei einer matrizenähnlichen Anordnung, einem zweidimensionalen Feld, sind für den Zugriff zwei Indizes erforderlich. Analog sind auch mehrdimensionale Felder mit mehreren Indizes möglich.

6.1. Ein- und mehrdimensionale Felder

In C werden (eindimensionale) Felder durch Angabe der Dimension hinter dem Identifikator, der im weiteren ein Feld bezeichnen soll, deklariert. Die Dimension sowie der Index zum Zugriff auf die Feldelemente werden in eckigen Klammern angegeben. Der Index wird – wie in der Systemprogrammierung – ab Null beginnend gezählt.

Beispiel:

```
char text [4];
text [0] ='a';
text [1] ='b';
text [2] ='c';
text [3] ='\0';
```

Es wird das Feld **text**, bestehend aus vier Elementen vom Typ **char**, vereinbart. Den vier Feldelementen werden anschließend ASCII-Zeichen zugewiesen.
Bei mehrdimensionalen Feldern werden die Dimensionen und die Indizes jeweils getrennt in eckigen Klammern angegeben. Die Anordnung der Feldelemente im Speicher erfolgt zeilenweise, das heißt, beim Übergang von einem Element zum nächsten ändert sich der letzte Index am schnellsten (Umgekehrt als in FORTRAN!).

Beispiel:

```
static int A [2] [3] =
{{2,3,5}, {1,4,6}};
```

$A_{0,0}$	2
$A_{0,1}$	3
$A_{0,2}$	5
$A_{1,0}$	1
$A_{1,1}$	4
$A_{1,2}$	6

Bild 1 Anordnung der Feldelemente im Speicher

Es wird ein zweidimensionales Feld vom Typ **int** vereinbart, wobei der erste Index die Werte 0 und 1 und der zweite Index die Werte 0, 1 und 2 annehmen kann. Den einzelnen Elementen werden Anfangswerte zugewiesen. Die Anordnung der Feldelemente mit ihren Anfangswerten ist in Bild 1 dargestellt.
Die Summe aller Feldelemente kann durch folgende Anweisung berechnet werden:

```
for (s=i=0;i<2;i++)
for (j=0;j<3;j++) s=A [i] [j];
```

Hinweis:

In C sind nur Operationen für die einfachen Datentypen definiert, das heißt, es gibt keine Operationen für ein ganzes Feld, sondern nur für die Feldelemente, sofern die Feldelemente einfache Datenobjekte sind.

6.2. Zeiger auf Felder

Zeiger werden oft zur Verarbeitung von Feldern benutzt. Zeiger können auf den Feldanfang und auf einzelne Feldelemente gesetzt werden. In C ist der Feldname identisch mit der Adresse des Feldes und entspricht somit der Adresse des nullten Feldelementes. Die Adresse muß über den Adreßoperator (**&**) gebildet werden.

Beispiel:

```
int A [10], *a,i;
a=A;
```

Der Zeiger **a** wird auf das Feld **A** gesetzt. Da **A [0]** das nullte Feldelement ist, führt die folgende Anweisung

a=&A [0];

zum gleichen Resultat. Mit der Anweisung

a=&A [i];

wird der Zeiger **a** auf die Adresse des i-ten Feldelementes gesetzt.

Hinweis:

Es besteht ein sehr enger Zusammenhang zwischen der Indizierung und der Zeigerarithmetik. Der Compiler verwandelt einen Verweis auf ein Feld in einen Zeigerwert. Dadurch ist ein Feldname immer ein Zeigerausdruck (siehe **a=A;**), und der Adreßoperator (**&**) ist in diesem Sonderfall nicht erforderlich.

Der Zugriff auf die einzelnen Feldelemente erfolgt über den Indirektoperator (*). Da es keine Operationen gibt, die sich auf Felder im ganzen beziehen, kann der Feldname als Zeiger auf den Feldanfang benutzt werden.

Beispiel:
a=A;
b=*a; oder **b=*A;**
c=*(a+i); oder **c=*(A+i);**

Der Variablen **b** wird der Wert des Elements **A [0]** und der Variablen **c** der Wert des Elements **A [i]** zugewiesen. Die Schreibweise ***(A+i)** ist somit ein Äquivalent für die indizierte Schreibweise **A [i]**.

6.3. Zeigerfelder

Mehrere Zeiger können zu Zeigerfeldern zusammengefaßt werden.

Beispiel:
int *a [4], A [10];

Das Feld **a** ist ein Zeigerfeld vom Typ **int**, bestehend aus vier Zeigern.

a [0] =&A [0];
a [1] =&A [2];
a [2] =&A [5];
a [3] =&A [9];

Den vier Zeigern werden die Adressen der Elemente **A[0], A[2], A[5]** und **A[9]** zugewiesen.

Die Zeiger des Zeigerfeldes können durch Angabe des Index verwendet werden.

Beispiel:
b=*a[0] + *a[2] − *a[3];
Der Ausdruck ist dem Wert von **A [0] + A [5] − A [9]** äquivalent.

Hinweis:

Entsprechend dem rekursiven Grundprinzip der Programmiersprache C können Zeiger auf Zeigerfelder gesetzt werden, die wiederum zu Zeigerfeldern zusammengefaßt werden können.

6.4. Zeichenfelder und Zeichenketten

Auch Zeichen können in Feldern angeordnet werden. Jedes Element eines Zeichenfeldes ist dann ein ASCII-Zeichen. In C gibt es keinen Zeichenkettentyp. Deshalb werden Zeichenketten als Zeichenfelder gespeichert. Trotzdem gibt es einen wichtigen Unterschied zwischen Zeichenketten und Zeichenfeldern, denn Zeichenketten müssen in C immer mit einem Nullzeichen **\0** enden. Bei der Definition von Zeichenkettenkonstanten wird vom Compiler das Nullzeichen **\0** automatisch angefügt. Somit sind **'1'** und **"1"** in C verschiedene Definitionen! Die Länge der Zeichenkette im Speicher ist daher immer um ein Zeichen größer. Auf Zeichenkettendefinitionen wird wie bei Feldern über Zeiger zugegriffen. Der Zeiger auf eine Zeichenkette verweist immer auf das erste Zeichen der Zeichenkette.

Beispiel:
char * text1;
text1 = "Ende des Programms TEXT1\n";

Vom Compiler wird die Zeichenkette als Konstante im Hauptspeicher abgelegt. Die Adresse der Zeichenkette wird dem Zeiger **text1** zugewiesen. Die Zeichenkette wird also nicht kopiert, denn an der Zuweisungsoperation sind nur Zeiger beteiligt.

Hinweis:

Die Programmiersprache C hat keine Operatoren, die eine Zeichenkette als ein Objekt behandeln, da es nur Operationen für die einfachen Datentypen gibt.

6.5. Initialisierung von einfachen Variablen und Feldern

Einfache Variablen können bei ihrer Definition grundsätzlich auch initialisiert werden. Der Variablen folgen dabei ein Gleichheitszeichen und ein Ausdruck. Im Ausdruck dürfen nur Variablen verwendet werden, die bereits definiert sind. Der Ausdruck darf in geschweifte Klammern eingeschlossen sein.

Beispiel:
static int a=20, b={(a+1)/2};

Variablen der Speicherklasse **static** und globale Variablen werden einmalig vom Compiler initialisiert. Variablen der Speicherklassen **register** und **auto** werden normalerweise nicht initialisiert. Manche Compiler lassen trotzdem eine Initialisierung zu, die bei jedem Blockeintritt wie eine Laufzeitanweisung ausgeführt wird.
Nicht initialisierte Variablen der Speicherklasse **static** und globale Variablen werden mit 0 initialisiert. Nicht initialisierte Variablen in den Speicherklassen **register** und **auto** haben einen beliebigen, in der Regel nicht reproduzierbaren Wert.
Ist das zu initialisierende Objekt ein Feld, dann besteht die Initialisierung aus einer Liste von Initialisierungen für die einzelnen Feldelemente. Die Liste ist in geschweifte Klammern einzuschließen, wobei die einzelnen Werte durch Komma getrennt werden. Die Initialisierung erfolgt in der Reihenfolge der Anordnung der Feldelemente. Wenn nicht genügend Werte angegeben werden, so wird der Rest des Feldes mit 0 initialisiert. Bei mehrdimensionalen Feldern kann die Initialisierung zeilenweise erfolgen, wobei jede Liste auch getrennt in geschweifte Klammern eingeschlossen werden kann. Felder der Speicherklasse **auto** können nicht initialisiert werden.

Beispiele:
```
static int a[10] = {1,2,5,7,10};
static int b[3] [4] = {{1,2,5,7},
                       {3,8,12,17},
                       {10,2,3,12}};
oder
static int b[3] [4] =
{1,2,5,7,3,8,12,17,10,2,3,12};
static int c[3] [4] =
{{1}, {2}, {3}};
```

Das Feld **a** wird mit 5 Werten initialisiert. Die restlichen Elemente werden auf 0 gesetzt. Die zwei Schreibweisen für die Initialisierung des Feldes **b** sind identisch. Im Feld **c** wird nur die erste Spalte initialisiert. Die restlichen Feldelemente werden auf 0 gesetzt.
Bei der Initialisierung von Feldern ist die Dimensionsangabe nicht zwingend er-

forderlich. Die Dimensionsangabe wird dann aus der Anzahl der Werte der Initialisierungsliste berechnet. Von dieser Möglichkeit wird auch bei der Initialisierung eines Zeichenfeldes mit einer Zeichenkette Gebrauch gemacht. Die geschweiften Klammern können bei der Initialisierung mit einer Zeichenkette entfallen.

Beispiel:
int d[] = {3,7,9,11};
char text[] = "Montag";
oder
char text [] =
{'M', 'o', 'n', 't', 'a', 'g', '\0'};

Die Verwaltung von mehreren Zeichenketten in einem Feld erfolgt zweckmäßigerweise in einem Zeigerfeld, das mit den Adressen der Zeichenketten initialisiert werden kann.

Beispiel:
```
static char * tag = {
                "",
                "Montag",
                "Dienstag",
                "Mittwoch",
                "Donnerstag",
                "Freitag",
                "Sonnabend",
                "Sonntag"
                    };
```

7. Strukturen und Vereinigungen

Eine Struktur (**struct**) ist ein Datenobjekt aus einer Folge von einzelnen benannten Komponenten, die Datenobjekte verschiedener Datentypen repräsentieren können. Der Speicherbereich einer Struktur wird vom Compiler aus der Summe der Speichergrößen der einzelnen Komponenten berechnet.
Eine Vereinigung (**union**) ist dagegen ein Datenobjekt, das auf einen virtuellen Speicherbereich Komponenten verschiedener Datentypen vereinigt. Die verschiedenen Komponenten können jedoch den Speicherbereich nur nacheinander belegen. Der Speicherbereich einer Vereinigung wird vom Compiler der größten Komponente entsprechend festgelegt.
Strukturen und Vereinigungen werden analog bereinbart und benutzt. In Strukturen und Vereinigungen können weitere Strukturen und Vereinigungen enthalten sein.

Strukturen und Vereinigungen werden wie folgt definiert:

struct [*name*] [{*member* [*member*...]}]
[*identifier* [,*identifier*...]];
union [*name*] [{*member* [*member*...]}]
[*identifier* [,*identifier*...]];

wobei
name – wahlfreier Name zur Bezeichnung der Struktur oder Vereinigung (Strukturname oder Vereinigungsname),
member – Spezifikation einer oder mehrerer Variablen eines bestimmten Datentyps und
identifier – Identifikator für eine Strukturvariable oder Vereinigungsvariable darstellen.

7.1. Definition von Strukturen und Vereinigungen

Werden in einer Struktur- bzw. Vereinigungsvereinbarung keine Identifikatoren angegeben, so wird lediglich die Struktur bzw. Vereinigung definiert. Es wird ein Name festgelegt, über den auf die Struktur- bzw. Vereinigungsvereinbarung Bezug genommen werden kann. Es erfolgt keine Vereinbarung von Struktur- bzw. Vereinigungsvariablen und somit auch keine Speicherplatzreservierung.

Beispiele:
struct art {char * nr; float pr;
int n;};
struct datum {int tag; char *
monat, jahr [2]};

Die erste Vereinbarung definiert eine Struktur **art** zur Beschreibung eines Artikels, bestehend aus einem Zeiger auf eine Zeichenkette zur Angabe einer Artikelnummer, einer **float**-Variablen für den Preis und einer **int**-Variablen für eine Stückzahl. Die zweite Vereinbarung definiert eine Struktur **datum**, die drei Komponenten enthält; eine **int**-Variable **tag**, einen Zeichenkettenzeiger **monat** und ein Zeichenfeld **jahr** aus zwei Zeichen.
Über die Namen **art** und **datum** kann im weiteren auf die definierten Strukturen Bezug genommen werden.

7.2. Vereinbarungen von Struktur- und Vereinigungsvariablen

Werden nach der Struktur- bzw. Vereinigungsvereinbarung Identifikatoren spezifiziert, so bezeichnen diese Identifikatoren im weiteren Struktur- bzw. Vereinigungsvariablen, deren Speicherobjekt die entsprechende Struktur bzw. Vereinigung realisiert.

Beispiele:
struct {char * nr; float pr;
int n;} a, b;
struct {int tag; char * monat,
jahr [2]} d1, d2, d3;

Es werden zwei Strukturvariablen **a** und **b** bzw. drei Strukturvariablen **d1**, **d2** und **d3** vereinbart, die jeweils ein Speicherobjekt der angegebenen Strukturen repräsentieren. Die Strukturen selbst sind unbenannt. Die Speicherstruktur der Strukturvariablen **a** ist in Bild 2 dargestellt.

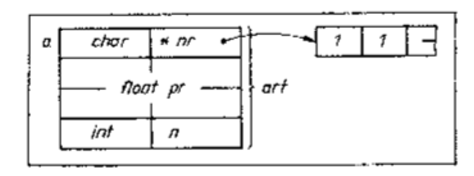


Bild 2 Beispiel für den Aufbau einer Struktur

Falls die Struktur bzw. Vereinigung bereits gemäß 7.1. definiert wurde, so kann die Vereinbarung der Struktur- bzw. Vereinigungsvariablen durch Bezugnahme auf den Namen der Struktur bzw. Vereinigung vereinfacht werden.

Beispiele:
struct datum d1, d2, d3;

Es werden drei Strukturvariablen **d1**, **d2** und **d3** für die Struktur **datum** vereinbart.

Entsprechend den Abschnitten 5. und 6. können auch Felder von Strukturen und Vereinigungen bzw. Zeiger auf Strukturen und Vereinigungen vereinbart werden.

Beispiele:
struct art a, b, t[20], * p;
struct datum d1, d2, d3,
tlist [200], * t1;

Das Feld **t** besteht aus 20 Feldelementen, wobei jedes Element die Struktur **art** realisiert.
Das Feld **tlist** besteht aus 200 Feldelementen, wobei jedes Element eine Struktur **datum** verkörpert.

RAM-Floppy – ein schneller Zusatzspeicher für Bürocomputer

Dr. Christian Löber
VEB Robotron-Meßelektronik
„Otto Schön" Dresden

Anwendung und Eigenschaften von RAM-Floppies

In Arbeitsplatz-, Personal- und Bürocomputern sowie in Entwicklungssystemen werden als Hintergrundspeicher vorrangig Diskettenspeicher eingesetzt. Durch weitere Verringerung der Abmessungen und Verbesserung der technischen Parameter gilt dies zunehmend auch für Heim- und Kleincomputer und darüber hinaus für intelligente Meßgeräte und Prüfsysteme wie etwa Digitaltester, Meßtechnikcontroller oder Funktionsmeßplätze. Nachteilig ist jedoch die niedrige Transfergeschwindigkeit der Floppy-Laufwerke. Dadurch wird die Effektivität des Bedieners bei üblichen Programmierarbeiten wie Laden, Editieren, Assemblieren, Compilieren, Linken von Programmen sowie beim Arbeiten mit Datenbank- und Textverarbeitungssystemen deutlich reduziert. Ganz besonders gilt das für Rechner und Geräte, die unter dem weit verbreiteten Betriebssystem CP/M bzw. dem dazu kompatiblen SCP arbeiten. Diese Betriebssysteme sind durch ihr streng hintergrundspeicherorientiertes Entwurfsprinzip, durch die angewandte Overlaytechnik ihrer Dienstprogramme sowie regelmäßige Warmstarts und damit sehr häufige Um- und Nachladeprozesse von bzw. zum Diskettenspeicher charakterisiert.
Davon ausgehend und unter Berücksichtigung des rasch steigenden Integrationsgrades sowie stark fallender Preise bei dynamischen Halbleiterspeichern (DRAM) werden in zunehmendem Maße sogenannte RAM-Floppies (RAM-Disk, virtuelle Disk) zur Geschwindigkeitssteigerung eingesetzt. Dies sind vollelektronische Zusatzspeicher, meist in mehreren Speicherbänken zu je 64 KByte organisiert, die sich aus der Sicht des Bedieners wie normale Diskettenspeicher verhalten, mit einem Unterschied: Sie sind um den Faktor 3 . . . 10 schneller /1, 2, 3/.
RAM-Floppies sind vor ihrem Gebrauch wie physische Disketten entsprechend den im jeweiligen Betriebssystem gültigen Konventionen zu initialisieren. Danach sind sie betriebsbereit und können mit häufig benutzten Dienstprogrammen geladen werden. Soll zum Beispiel ein Anwenderprogramm auf Assemblerniveau erstellt werden, so wird man den Editor oder ein Textverarbeitungsprogramm kopieren, ferner den Assembler, Linker und ggf. – für anschließendes Testen – einen geeigneten Debugger. Damit sind alle für Programmierung und Test benötigten Dienstprogramme auf der RAM-Floppy verfügbar, und alle Diskettenoperationen verlaufen mit einer um den oben genannten Faktor höheren Geschwindigkeit.
Nach Abschluß der Arbeiten, beispielsweise nach dem Ausdrucken des in der RAM-Floppy stehenden Listings, muß als Besonderheit beachtet werden, daß die erstellten Daten für spätere Verfügbarkeit auf eine physische Diskette auszulagern sind, ansonstens gehen sie beim Ausschalten des Rechners verloren.

Wirkungsprinzip von RAM-Floppies

Es gibt zwei prinzipielle Möglichkeiten, um eine RAM-Floppy in ein bestehendes System zu integrieren:

I/O-Mapping

Hierbei wird die aus einer oder mehreren Seiten bestehende RAM-Floppy über hard- oder softwaregesteuerte, bitserielle oder bitparallele I/O-Ports an das Grundsystem angeschlossen. Damit ist die RAM-Floppy ein echter vollelektronischer Zusatzspeicher, der nicht im Adreßraum des Hauptspeichers, sondern in dem des I/O-Raumes liegt. Der Datentransfer wird durch Blockein-/-ausgabebefehle – beim U880 ist das die Gruppe der INIR/OTIR-Befehle – realisiert. Das bedeutet bei 8-Bit-Rechnern, daß nur maximal 128 Byte lange Sektoren in einem Zug transferiert werden können. Darüber hinaus muß bei dieser Lösung das Problem des zyklischen Refresh der DRAM-Speicher beachtet werden. In /1/ ist eine Schaltung für eine halbintelligente Diskettenstation mit einer RAM-Floppy beschrieben, die aus Geschwindigkeitsgründen bitparallel und rein hardwaregesteuert über einen IEC-Controller arbeitet. Eine andere, in /4/ erläuterte Variante basiert hingegen auf einfachen richtungsgesteuerten Datentreibern und einer fünfstufigen, im Bereich von 0 . . . FFFF beliebig einstellbaren Zählkette, die das Speichersegment adressiert, auf das zugegriffen werden soll.

Memory-Mapping

Bei diesem Prinzip wird die RAM-Floppy im allgemeinen in Bänken zu je 64 KByte mit dem Hauptspeicher „gemappt", das heißt, sie liegt voll in seinem Adreßraum. Zur Vermeidung von Adressenkonflikten ist deshalb eine Seitenschaltung erforderlich, die die zur aktuellen physischen Diskettenadresse (DRIVE, TRACK, SECTOR) gehörende Speicherseite an die Stelle der weggeschalteten Grundseite des Hauptspeichers über Speichersperr-/-freigabe-Signale (wie z. B. MEMDI, MEMDI1, MEMDI2 beim Rechner K 1520) einblendet. Der Datentransfer wird programmseitig mit Blocktransferbefehlen, wie z. B. LDIR, realisiert. Damit können maximal 64 KByte lange Sektoren in einem Zug kopiert werden. Ein Beispiel für das Memory-Mapping-Prinzip ist in /2/ beschrieben, wobei 32 KByte lange Halbseiten in den Adreßbereich 4000 . . . BFFF der Grundseite eingeblendet werden. Die Variante nutzt die nicht für alle Betriebssysteme geltende Eigenschaft von CP/M, daß jeder Datenaustausch sektorweise zwischen Grundseite und Diskettenspeicher über einen globalen DMA-Puffer außerhalb des ausblendbaren Bereiches 4000 . . . BFFF erfolgt. Damit ist ein Diskettenzugriff in zwei voneinander unabhängigen Schritten realisierbar: beispielsweise beim Lesen der Diskette zuerst Transfer eines Sektors zwischen Diskette und BIOS-internem DMA-Puffer und danach Transfer zwischen diesem DMA-Puffer und dem adressierten Segment im Bereich 4000 . . . BFFF. Wird vor dem zweiten Schritt durch die Seitensteuerung eine der Halbseiten einer RAM-Floppy eingeblendet, so ist dies ein Kopiervorgang von einer physischen Diskette auf die RAM-Floppy.
Die Voraussetzung einer Zwischenpufferung der zu transferierenden Daten in einem DMA-Puffer ist jedoch für 8-Bit-Betriebssysteme nicht allgemein üblich, so zum Beispiel nicht bei dem Betriebssystem UDOS bzw. dem dazu kompatiblen RIO.

Die erläuterten Hardware-Erweiterungen für eine RAM-Floppy müssen außerdem softwareseitig unterstützt werden:

- Deklarierung des zusätzlichen virtuellen Laufwerks im Betriebssystem. Dies umfaßt in der Regel die Ergänzung/Erweiterung der vorhandenen Diskbeschreibungsvektoren.
- Erweiterung des Betriebssystems um einen RAM-Floppy-Treiber, der – ähnlich wie der „normale" Floppy-Treiber die physische Diskettenarbeit – die virtuelle Diskettenarbeit realisiert. Dies erfordert u. a. bei einem konkreten

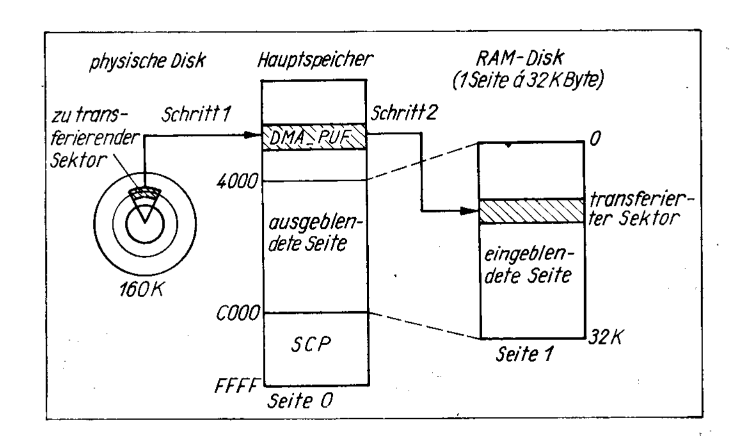


Bild 1 Transfer eines 128-Byte-Sektors in zwei Schritten von der physischen Diskette auf die eingeblendete zweite Seite der virtuellen Diskette

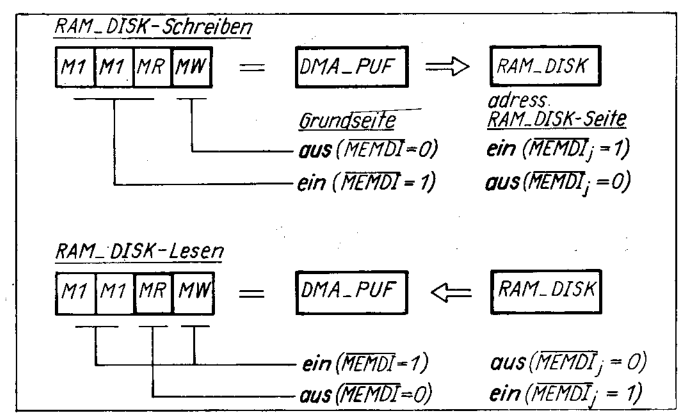


Bild 3 Zyklusgesteuerte Seitenumschaltung während RD/WR-Zyklus des LDIR-Blocktransferbefehls ($\overline{MEMDI}$ = 1: adressierte RAM-Floppyseite eingeschaltet)

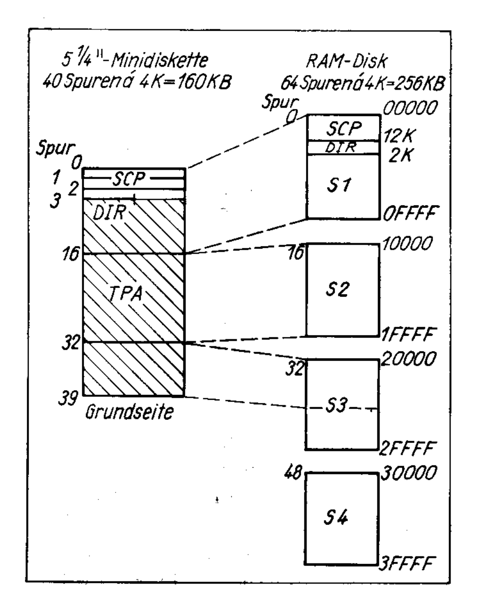


Bild 2 Organisation einer 5¹/₄"-Minidiskette mit 40 Spuren und einer 256-K-RAM-Floppy unter SCP (Spur 0, 1, 2: SCP; halbe Spur 3: Directory DIR; Spur 3...39/64) TPA – transient programm area (Anwenderprogramme)

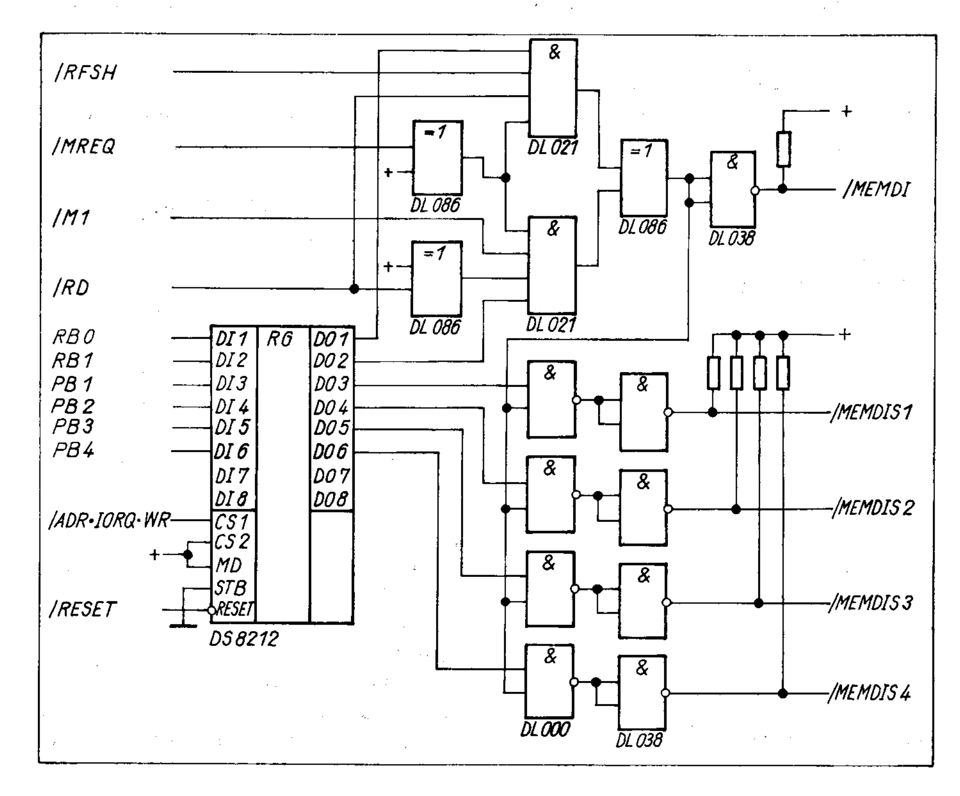


Bild 4 Seitenschaltung für RAM-Floppy nach dem MEMORY-MAPPING-Prinzip

Diskettenzugriff die Konvertierung der bereitgestellten physischen Diskettenadresse in die korrespondierende Adresse auf der RAM-Floppy sowie die Modifikation der Systemrufe des Betriebssystems, die das physische Schreiben/Lesen der adressierten Diskette ausführen.

Eine RAM-Floppy für den Bürocomputer A 5120

Im folgenden wird ein Beispiel zur Implementierung einer RAM-Floppy in den Bürocomputer A 5120 nach dem Memory-Mapping-Prinzip erläutert. Dabei umfaßt die RAM-Floppy 4 Seiten zu je 64K = 256 KByte. Sie repräsentiert damit die 1,6fache Kapazität der derzeit üblichen 5¹/₄"-Minidisketten doppelter Dichte mit 40 Spuren zu je 4 KByte. Bild 2 zeigt die Organisation einer solchen Diskette unter SCP. Demzufolge ist bei jedem Zugriff auf das virtuelle Floppy-Laufwerk durch den zugehörigen RAM-Floppy-Treiber über die Seitenschaltung folgende Adressenkonvertierung zwischen physischer Diskadresse und RAM-Diskadresse zu realisieren:

physische Diskette	RAM-Floppy
Spur 0...15	00000...0FFFF (Seite 1)
Spur 16...31	10000...1FFFF (Seite 2)
Spur 32...47	20000...2FFFF (Seite 3)
Spur 48...63	30000...3FFFF (Seite 4)

Die zusätzlich erforderlichen Adreßbits zur Adressierung des Speichers der RAM-Floppy werden hierbei über vier Pagingbits PB 1...4 eines Ports in der Seitenschaltung simuliert. So ist beispielsweise bei einem Sektortransfer im Bereich von Spur 16...31 durch den RAM-Floppy-Treiber vor dem Blocktransferbefehl LDIR als statische Bedingung für die Seitensteuerung das Bit PB 2 zu setzen.
Darüber hinaus ist noch die Richtung des Datentransfers von bzw. zur virtuellen Diskette festzulegen (RAM-Floppy-Lesen/Schreiben). Dies geschieht durch zwei Richtungssteuerbits RB0 und RB1, die ebenfalls über Port im entsprechenden READ- bzw. WRITE-Ruf des SCP statisch einzustellen sind.

```
                 ;*************************************************************
                 ;***                                                       ***
                 ;***       UP DISKRW:   RAM_DISK-Lesen/Schreiben           ***
                 ;***                                                       ***
                 ;*************************************************************
                 ;file:  DSKRDWR.MAC
                 ;date:  01.09.86_LO
                 ;
0080             PagPrt  equ    80h          ;Port f.Ausg.Pagbits PB1..4
                                             ;u.Richtungssteuerbits RB0/1
                         .z80
0000'                    aseg
                         org    100h
                 ;
0100             DISKRW:
                 ;       ...                 ;Steuerwort SW/akt.Seite berech.
                 ;       ...                 ;z.B. 00000101 ==) RDISK_WR/S.1
                 ;
0100    F3               di
0101    D9               exx                 ;Transferbed.in Zweitreg.def.
0102    21 011C          ld     hl.Quelle
0105    11 011E          ld     de.Ziel
0108    01 0080          ld     bc.80h       ;1 Sec.transferieren
010B    D9               exx
010C    3A 0120          ld     a.(SW)
010F    0E 80            ld     c.PagPrt     ;Port Adr.
                 ;-------------------------------------------------------------
0111    ED 79            out    (c).a        ;akt.Seite einschalten
0113    D9               exx
0114    ED B0            ldir                ;Transf.DMA_Puff ==) RAM DISK
0116    D9               exx
0117    AF               xor    a            ;SW:= 00000000
0118    ED 79            out    (c).a        ;Seitensteuerung aus
                 ;-------------------------------------------------------------
011A    FB               ei
011B    C9               ret
                 ;
011C    0000     Quelle: dw     00           ;Sector-Quelle
011E    0000     Ziel:   dw     00           ;Ziel-Sector
0120    00       SW:     db     00000000     ;St.wort: xxPB4PB3PB2PB1RB1RB0
                 ;
                 ;*************************************************************
```

Bild 5 Programm DISKRW

Dynamische Seitensteuerung

Die eigentliche Seitenumschaltung erfolgt jedoch dynamisch erst während der Abarbeitung des LDIR-Befehls und nur für die Dauer eines READ- bzw. WRITE-Zyklus. Das bedeutet, z. B. beim RAM-Floppy-Schreiben, daß das nächste Datenbyte im DMA-Puffer während des MR-Zyklus des LDIR-Befehls noch bei aktiver Grundseite gelesen wird. Erst mit der fallenden Flanke von $\overline{MREQ}$ im MW-Zyklus wird die Grundseite aus- und dafür die adressierte RAM-Floppy-Seite eingeblendet und somit das Byte in die RAM-Floppy geschrieben. Anschließend, das heißt, mit der aufsteigenden Flanke von $\overline{MREQ}$, erfolgt die Wiedereinschaltung der Grundseite. Die RAM-Floppy-Seite ist folglich nur für die Dauer von $\overline{MREQ} = 0$ eingeblendet. Bild 3 verdeutlicht die Verhältnisse.
In Bild 4 ist das Prinzip der Seitenschaltung dargestellt. Die einwandfreie Funktion der Schaltung ist an eine programmtechnische Besonderheit gebunden. In der Programmpassage des RAM-Floppy-Treibers dürfen nach dem Setzen eines der beiden Richtungssteuerbits RB0, RB1 unmittelbar vor dem LDIR-Befehl bis zu ihrem Rücksetzen unmittelbar nach dem LDIR-Befehl nur Befehle mit M1-Zyklen auftreten, mit Ausnahme des LDIR selbst. Ansonsten treten Fehlschaltungen der Seitensteuerung durch unbeabsichtigte MR- bzw. MW-Zyklen auf. Das in Bild 5 angegebene Programm DISKRW genügt diesen Bedingungen.
Die dynamische Umschaltung der Seiten nur im jeweils aktiven RD- bzw. WR-Zyklus des Blocktransferbefehls hat folgende Vorzüge:

- Der DMA-Puffer von SCP (über den Systemruf SETDMA durch den Systemprogrammierer frei wählbar) kann bei beliebigen Adressen im Bereich der gesamten Grundseite 0 ... FFFFH liegen, im Gegensatz z. B. zu der Lösung in /2/.
- Das Verfahren funktioniert auch bei Betriebssystemen, bei denen der Datentransfer direkt zwischen den aktiven Sektoren auf der Diskette und dem Hauptspeicher unter Umgehung von Zwischenpuffern erfolgt (z. B. RIO, UDOS).
- Es können – wie bei verschiedenen Betriebssystemen üblich – auch längere als nur 128-Byte-Sektoren in einem Zug transferiert werden.
- Durch den Blocktransfer mit Hilfe des LDIR-Befehls gibt es keine Refresh-Probleme.

RAM-Floppy-Treiber

Wie schon erwähnt, ist zur Abwicklung der Diskettenarbeit mit der RAM-Floppy ähnlich wie bei physischen Disketten ein RAM-Floppy-Treiber erforderlich. Er enthält u. a.:

Tafel 1

Pagingbit				Speichersperr/freigabesignal					aktive Seite
PB1	PB2	PB3	PB4	/MEMDI	/MEMDIS1	/MEMDIS2	/MEMDIS3	/MEMDIS4	
0	0	0	0	1	0	0	0	0	Grundseite
1	0	0	0	0	1	0	0	0	Seite 1/RAM-DISK
0	1	0	0	0	0	1	0	0	Seite 2/RAM-DISK
0	0	1	0	0	0	0	1	0	Seite 3/RAM-DISK
0	0	0	1	0	0	0	0	1	Seite 4/RAM-DISK

Tafel 2

Richtungs-Steuerbit		Funktion
RB1	RB2	
0	0	Grundzustand (Hauptspeicher aktiv)
0	1	RAM_DISK_SCHREIBEN
1	0	RAM_DISK_LESEN
1	1	unzulaessig

Tafel 3

Operation	File		Reaktionszeiten/sec phys.Disk	Reaktionszeiten/sec RAM_DISK	Geschwindigkeitssteigerung
Starten von Dienstprogrammen	DU	(10k)	5	< 1 x	
	ASM	(16k)	8	< 1 x	1 : 10
	REDABAS	(20k)	9	< 1 x	
	TP	(32k)	16	2	1 : 8
Kopieren von Datein			phys./phys.	phys/RDISK	
	POWER	(16k)	21	7	
	REDABAS		27	8	1 : 3
	TP		39	13	
Assemblieren	DRIVEMUX.MAC	(6k)	80	14	1 : 5,7
	BIOS722.MAC	(32k)	310	90	1 : 3,4
	(+ 2 INCLUDE-files)				
Linken	DRIVEMUX.REL	(2k)	12	3	
	BIOS722.REL	(6k)	30	8	1 : 4
typ.REDABAS-Funktionen	Datei sortieren xx		47	16	1 : 3
	Datei indizieren xx		55	9	1 : 6
	Datei sortieren xxx		556	62	1 : 9
Warmstart			5...7	< 1 x	> 1 : 10

x augenblickliche Reaktion
xx Sortieren/Indiz.einer Namensliste (107 Zeilen a 21 Spalten,124 Zeich.lang)
xxx Sortieren einer Stueckliste (700 Positionen a 14 Felder,131 Zeichen lang)

– einen Drive-Multiplexer als Weiche zwischen physischen und RAM-Floppy-Zugriffen (bei SCP erfordert dies eine Änderung der Rufe READ und WRITE im BIOS).
– ein Konvertierungsprogramm zur Umrechnung der physischen Diskadresse (in SCP definiert durch die BIOS-Schnittstelle DRIVE, TRACK, SECTOR) in die zugehörige RAM-Floppy-Adresse.
– den eigentlichen „Treiber" zur Realisierung des sektorweisen Datentransfers in Verbindung mit der erläuterten statischen bzw. dynamischen Seitensteuerung.
– Fehlerbehandlung und Realisierung von Sonderfunktionen (z. B. Read-after-Write-Funktion zur Erkennung von fehlerhaften Schreiboperationen auf der RAM-Floppy).

Er ist in das jeweilige Betriebssystem einzubinden, und darüber hinaus sind die Diskettenbeschreibungstabellen zu modifizieren. Unter SCP bedeutet dies die Eröffnung eines Diskparameterblockes für die RAM-Floppy. Dieser beschreibt dann deas Format der RAM-Floppy, die hinsichtlich Speicherkapazitäten, Anzahl der Systemspuren usw. nicht notwendigerweise mit dem Format der physischen Diskette übereinstimmen muß.

Praktische Ergebnisse

Nach dem vorgegebenen Prinzip wurde eine 256-K-RAM-Floppy, bestehend aus dem eigentlichen RAM-Floppy-Treiber und einem 256-K-DRAM-Speicher mit 4 Bänken zu je 64 KByte, realisiert und im Bürocomputer A 5120 eingebaut. Das Betriebssystem SCP wurde in einer in seinen Grundkomponenten CCR, BDOS und BIOS entsprechend modifizierten und erweiterten Version SCPNET V1.2 implementiert. Die aus der Literatur und theoretischen Betrachtungen erwartete Verkürzung der Reaktionszeiten hat sich in jeder Hinsicht bestätigt. Die bei typischen Programmieraktivitäten gemessene Geschwindigkeitssteierung bewegt sich im Verhältnis 1 : 3 bis 1 : 10, das heißt, die unangenehmen Wartezeiten haben sich drastisch verringert (Tafel 3). Die Geschwindigkeitsvorteile äußern sich speziell beim iterativen Prozeß der Programmerstellung von Mikrorechnersoftware auf Grund des notwendigen häufigen Programmwechsels, aber auch bei der Arbeit mit Datenbanksystemen und bei der Textverarbeitung in einem deutlichen Effektivitätszuwachs des Programmierers. Darüber hinaus wird das Arbeiten mit der RAM-Floppy durch die in vielen Fällen augenblickliche Reaktion des Systems und den damit verbundenen Tempogewinn auch subjektiv als sehr angenehm empfunden.
Für die praktische Arbeit mit der RAM-Floppy sind noch einige Sonderfunktionen von Bedeutung:
■ Die RAM-Floppy wird beim Netzeinschalten automatisch initialisiert, jedoch nicht bei RESET, so daß auf der RAM-Floppy vorhandene Dateien nicht gelöscht werden.
■ Der Warmstart als eine wesentliche Komponente der Geschwindigkeitssteigerung wurde auf das Laufwerk J der RAM-Floppy verlagert. Das Nachladen der Betriebssystemkomponenten CCP und BDOS erfolgt von diesem und nicht mehr vom Laufwerk A aus.
■ Schreiboperationen auf die RAM-Floppy werden durch die mitimplementierte RD-after-WR-Funktion überwacht. Bei auftretenden Schreibfehlern meldet sich das System mit der typischen laufwerksabhängigen Ausschrift „SCPX ERROR ON J: BAD SECTOR" zurück.

Das System ist gleichzeitig für den Anschluß an eine Festplatte über ein lokales Netz mit Lichtwellenleiterkopplungen vorbereitet. Dazu enthält das Betriebssystem SCPNET für die Arbeit mit virtuellen Disketten über die virtuellen Laufwerke D . . . I softwareseitige Erweiterungen.
Das System ist ab sofort nachnutzbar.

Literatur

/1/ Eisenack, G.; Fingberg, H: „RAM-Floppy"-Laufwerk beschleunigt den Zugriff. Elektronik (1985) 24, S. 85–90
/2/ Gassner, S.; Meier, R.: Halbleiterfloppy sind schneller . . . Elektrotechnik 65 (1983) 9, S. 19 bis 21
/3/ Joepgen, H.: Virtuelle Floppy. mc (1983) 9, S. 47–49
/4/ Sternberg, G.: Ein RAM-Floppy für den MC-CP/M Computer. mc (1985) 6, S. 86–96

✉ KONTAKT ☎

VEB Robotron-Meßelektronik „Otto Schön" Dresden, PSF 211, Dresden, 8012, Tel. 4 87 52 07, Dr. C. Löber

KC 85/2 als intelligentes grafisches Display für den PC 1715

Prof. Dr. Herbert Stuhec,
Dr. Dieter Vyhnal, Michael Rathmann
Ingenieurhochschule Köthen,
Sektion Anlagenbau

Die Kopplung des Personalcomputers PC 1715 /1/ mit einem Kleincomputer KC 85/2 bzw. KC 85/3 stellt eine Lösung dar, um mit dem in der Grundvariante nicht grafikfähigen Personalcomputer farbige grafische Darstellungen auf einem Bildschirm realisieren zu können. Der KC 85/2 /2/ ermöglicht

– die Darstellung vollgrafischer Bilder im Raster von 320 Punkten (in der Waagerechten) mal 256 Punkten (in der Senkrechten)
– die Zuordnung eines Farbbytes zu je einem Bildsegment von 8 × 4 Bildpunkten. Das Farbbyte ermöglicht die Auswahl von 16 Vorder- und 8 Hintergrundfarben für das Bildfeld.
– die blinkende Darstellung von 16 Vordergrundfarben im Bildfeld.

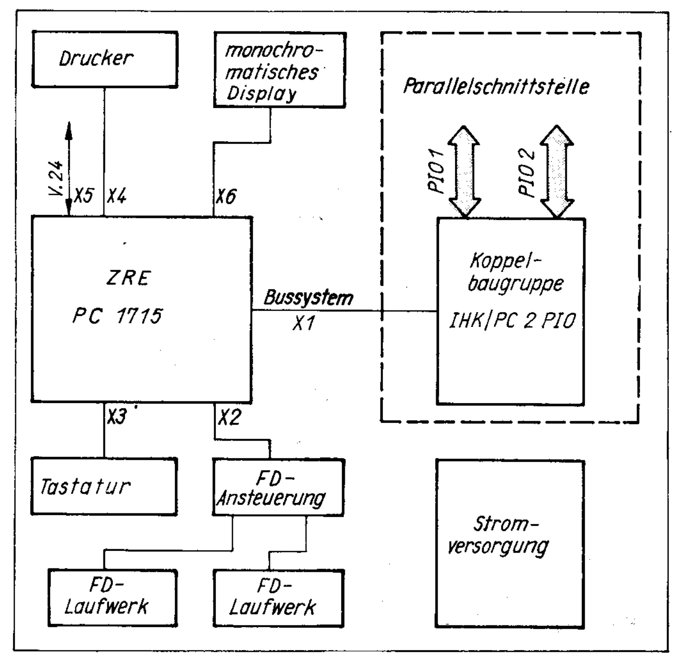
Durch die Kopplung PC 1715 – KC 85/2 können diese Farbgrafikfunktionen direkt am PC genutzt werden. Der KC 85/2 wird in Verbindung mit einem Farbmonitor zum intelligenten Farbgrafikdisplay. Die Realisierung ist mit geringem Hardwareaufwand möglich, da nur zwei Koppelbaugruppen eingesetzt werden müssen. Damit ist eine schnelle und unkomplizierte Nachnutzung des Konzeptes möglich.

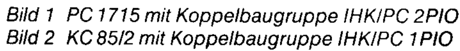
1. Systemgestaltung

Die Kopplung des PC 1715 mit dem KC 85/2 erfolgt durch eine Parallelschnittstelle, die über PIO-Ports an beiden Rechnern hardwaremäßig realisiert wird. Die Parallelschnittstelle ermöglicht den Datentransport zwischen den Rechnern mit hoher Geschwindigkeit und gestattet eine einfache Synchronisation.

1.1. Parallelschnittstelle für den PC 1715

Der Hersteller des PC 1715 bietet Leiterplatten zur Erweiterung der Kopplungsmöglichkeiten des Rechners an, die nicht zur Grundausstattung gehören. Es ist die Nachrüstung einer IFSS-Interfaceeinheit oder einer V.24-




Bild 1 PC 1715 mit Koppelbaugruppe IHK/PC 2PIO
Bild 2 KC 85/2 mit Koppelbaugruppe IHK/PC 1PIO

Interfaceeinheit möglich. Unter Nutzung des vorhandenen Bussystems X1 /3/ wurde eine neue Koppelbaugruppe mit Parallelinterface, IHK/PC 2 PIO, entwickelt, die anstelle einer der bereits erwähnten Interfacesteckeinheiten eingesetzt wird (Bild 1).

Die Koppelbaugruppe wurde mit 2 PIO-Schaltkreisen UB 855 bestückt. Da der X1-Bus des Rechners direkt genutzt wurde, ergaben sich folgende Toradressen:

Steuerregister	Datenregister
PIO 1:	
Kanal A 12h	Kanal A 10h
Kanal B 13h	Kanal B 11h
PIO 2:	
Kanal A 16h	Kanal A 14h
Kanal B 17h	Kanal B 15h

Zur Kopplung wird nur PIO 1 genutzt. Der zweite PIO-Schaltkreis steht für zusätzliche Parallel-Ein-Ausgaben zur Verfügung, z. B. für den Anschluß eines Analog/Digitalumsetzers oder einer CCD-Zeilenkamera.

1.2. Parallelschnittstelle für den KC 85/2

Für den KC wurde die Koppelbaugruppe IHK/KC 1 PIO entwickelt, die an das Expansionsinterface angeschlossen wird (Bild 2). Die Baugruppe besteht aus einem PIO-Schaltkreis UB 855 und der dazugehörigen Anschlußelektronik. Für die PIO-Ports wurden die Toradressen

PIO:	
Steuerregister	Datenregister
Kanal A CAh	Kanal A C8h
Kanal B CBh	Kanal B C9h

realisiert. Die Koppelbaugruppe kann als separate Baugruppe aufgebaut oder anstelle eines Moduls in einen freien Modulschacht des Rechners gesteckt werden.

1.3. Kopplung und Synchronisation

Die realisierte Kopplungs- und Synchronisationsvariante zum Anschluß des KC als intelligentes Ausgabegerät des PC ist im Bild 3 dargestellt.
Der Datentransfer zwischen den Rechnern wird interruptgesteuert durchgeführt. Nach der Ausgabe des Datenwortes und des Steuerwortes durch den PC wird ein Interrupt am KC 85/2 ausgelöst. Die Daten werden übernommen und verarbeitet. Es erfolgt die Ausgabe der grafischen Informationen auf dem Farbmonitor. Ist der KC 85/2 zur Übernahme eines weiteren Datenwort-Steuerwortpaares bereit, wird ein Interrupt am PC 1715 ausgelöst, der die Übertragung der nächsten Daten einleitet. Durch den interruptgesteuerten Datenaustausch zwischen PC 1715 und KC 85/2 wird eine optimale Datentransfergeschwindigkeit erreicht.

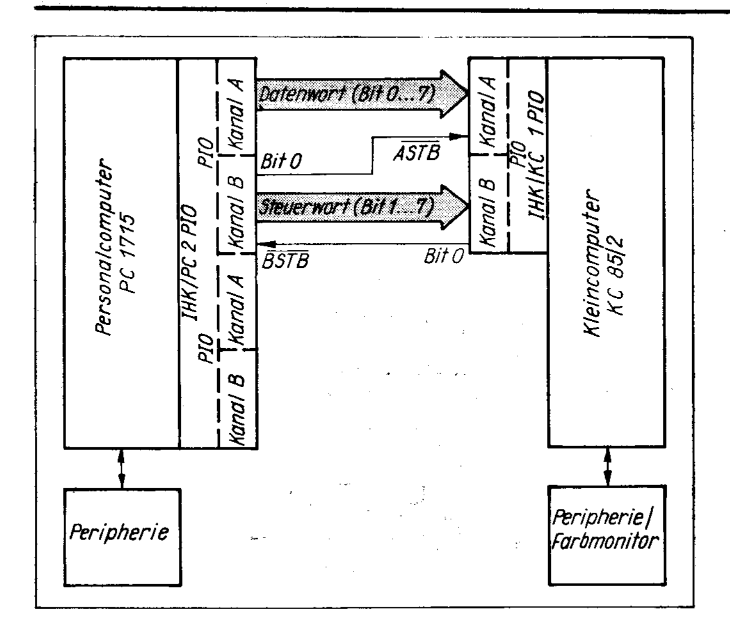
2. Software

Für die Nutzung des KC 85/2 als Farbgrafikdisplay des PC 1715 existieren prinzipiell zwei Möglichkeiten.
Einmal kann ein Frame-Buffer, d. h. die Bildpunktmatrix $P_{k,n} < P_{256,320}$ des Farbgrafikdisplays, im PC aufgebaut und verwaltet werden. Jeder Bildpunkt der Matrix wird in einem Byte des Arbeitsspeichers bzw. einer Bilddatei dargestellt.
Durch Grafikprogramme können Linien, Kreise, Kreissegmente und Text im Frame-Buffer generiert werden. Die Ausgabe der Bildpunktmatrix $P_{k,n}$, also des Frame-Buffers, erfolgt, nachdem die Generierung des Bildes abgeschlossen ist. Diese Variante bedingt einen hohen Softwareaufwand, da ein Frame-Buffer $P_{256,320}$ nicht komplett im Arbeitsspeicher des Rechners aufgebaut werden kann, ist aber, wie Abschnitt 3 zeigen wird, eine brauchbare Methode zur Darstellung von Binärbildern als Ergebnis einfacher Bildverarbeitungsprozesse. Eine weitere elegante Möglichkeit für Grafik- und Diagrammausgaben besteht darin, im PC 1715 ein Displayfile aufzubauen. Es wird eine Matrix $P_{g,l}$ erzeugt, deren Zeilen die Informationen Anfangspunktkoordinate, Endpunktkoordinate, Linie oder Kreisbogen, Farbe usw. enthalten. Jeweils eine Zeile der Matrix $P_{g,l}$ wird an den KC 85/2, der als intelligentes Grafikdisplay die Linien- und Kreisgenerierung, die Farbcodierung, die Transformation zur Bildschirmausgabe sowie die Schrifteinblendung selbständig übernimmt, übergeben. Die Ausgabe eines Steuerwortes von 7 Bit parallel zu den Daten realisiert 128 verschiedene Möglichkeiten der Funktionssteuerung des Farbgrafikdisplays. Beispielsweise sind die Funktionen

- direkte Ausgabe von Bildpunktmatrizen
- intelligenter Grafikmodus
- Bildschirmlöschen
- Farbbeeinflussung

möglich.

Bisher wurde eine Teilmenge der Vielfalt, die sich durch den Einsatz des KC 85/2 als intelligentes Farbgrafikdisplay ergibt, verwirklicht.



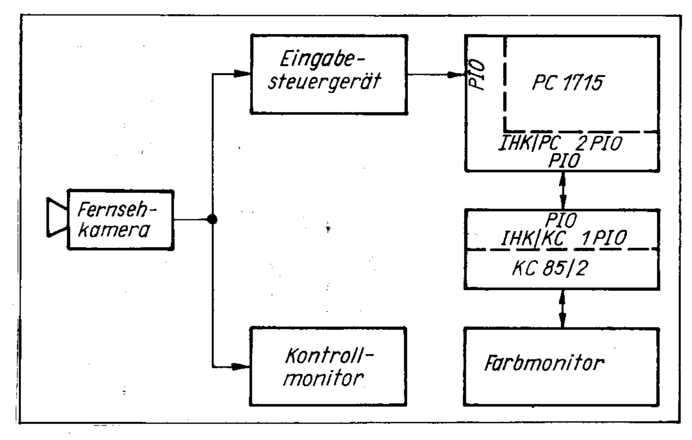


Bild 5 Vereinfachtes Bildverarbeitungssystem

Bild 3 Kopplung und Synchronisation

2.1. BILDOUT

Das Programm BILDOUT ermöglicht die Ausgabe einer optimierten Bildpunktmatrix $P_{256,256}$, die im Arbeitsspeicher des PC oder als Datenfile auf Disketten abgelegt ist, zum Farbgrafikdisplay. Es wird immer eine vollständige Matrix $P_{256,256}$ ausgegeben. Ist die reale Bildpunktmatrix kleiner, werden die fehlenden Werte so ergänzt, daß sie auf dem Farbgrafikdisplay die Hintergrundfarbe annehmen.
Die Elemente der Bildpunktmatrix werden im PC 1715 in einem Byte des Speichers dargestellt. Sie können Werte von 0 bis 255 annehmen. Vor der Datenübergabe erfolgt eine Datenreduktion. Jeder Punkt der Bildpunktmatrix wird durch Vergleich mit einer Triggerschwelle in einen binären Wert umgewandelt. Ist der Wert des Bildpunktes kleiner als die Triggerschwelle, wird dem Byte der Wert 0 zugeordnet, anderenfalls der Wert 1. Jeweils 8 Byte mit 0/1-Information werden dann zu einem Binärbyte zusammengefaßt. Jedes Bit des Binärbytes repräsentiert einen Bildpunkt. Über ein Steuerwort wird das Programm BILDKC auf der Grafikdisplayseite aktiviert.

2.2. BILDKC

Das Programm BILDKC realisiert die interruptgesteuerte Eingabe der Binärbytes. Die Daten werden direkt an den Grafikspeicher des KC 85/2 übergeben und auf dem Monitor dargestellt. Weitere Steuerworte gestatten das Setzen einer Vorder- und einer Hintergrundfarbe für das Bild.
Der KC 85/2 realisiert beim kontinuierlichen Füllen des Grafikspeichers keine kontinuierliche zeilenweise Ausgabe auf dem Bildschirm. Es werden die Zeilen 0, 4, 8, 12, dann die Zeilen 1, 5,

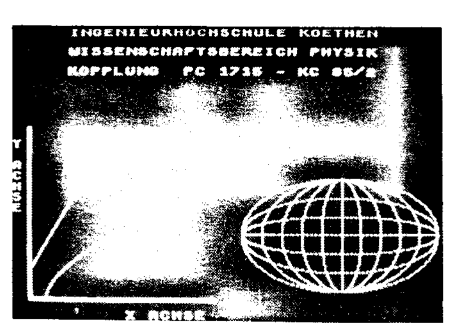


Bild 4 Testbild

9, 13 usw. geschrieben, bis ein Block zu 16 Zeilen dargestellt ist. 16 Blöcke ergeben das Gesamtbild. Durch entsprechende Adreßrechnung wird vom Programm BILDKC sichergestellt, daß die kontinuierlich übertragenen Zeilen der Bildpunktmatrix auch kontinuierlich auf dem Bildschirm dargestellt werden.

2.3. GRAFOUT

Das Programm GRAFOUT realisiert den Aufbau einer Datenmatrix $P_{g,l}$ im Arbeitsspeicher des Personalcomputers. Diese „Koordinatenmatrix", d. h. das Displayfile, kann auf Diskette abgespeichert oder zum Farbgrafikdisplay ausgegeben werden. Es besteht auch die Möglichkeit der direkten Ausgabe von der Diskette zum KC 85/2. Die Ausgabe der Matrix erfolgt zeilenweise, da jede Zeile die Koordinaten eines Punktes, einer Geraden, eines Kreises oder eines Schriftzeichens sowie entsprechende Zusatzinformationen enthält.
Das Programm GRAFOUT stellt die Daten für das BASIC-Programm GRAFIK bereit.

2.4. GRAFIK und GRAFKC

Das Assemblerprogramm GRAFKC ist im Farbgrafikdisplay, d. h. im KC 85/2, Bestandteil des BASIC-Programms

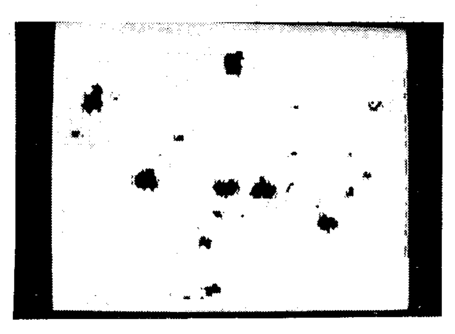


Bild 6 Optimales Binärbild; dargestellt auf dem Kleincomputer KC 85/2

GRAFIK. Der KC 85/2 wird mit einem Zusatzmodul M006 BASIC ausgerüstet. GRAFKC realisiert die interruptgesteuerte Eingabe jeweils einer Zeile der Datenmatrix $P_{g,l}$. Die Daten werden auf Arbeitsspeicherplätzen abgelegt. Mit Hilfe des BASIC-Befehls PEEK (I) werden jeweils nach der Eingabe einer Zeile die Arbeitsspeicherplätze gelesen und ihr Wert einer Variablenliste zugeordnet. Diese Variablenliste bildet die Grundlage zur Abarbeitung eines Grafikausgabebefehls /4/:

- LINE XA, YA, XE, YE, F
(Linie ziehen)
- CIRCLE XM, YM, R, F
(Kreis ziehen)
- PSET X, Y, F
(Punkt setzen)
- PRESET X, Y, F
(Punkt löschen)
- PAPER H
(Hintergrundfarbe setzen)
- INK V
Vordergrundfarbe setzen)
- COLOR V, H
(Vorder- und Hintergrundfarbe setzen).

Die Art des abzuarbeitenden Befehls ist in jeder Matrixzeile in einem Byte verschlüsselt. Nach dessen Ausführung wird wieder die Subroutine GRAFKC, die die Eingabe einer neuen Martixzeile realisiert, aufgerufen.

3. Anwendungen

Die Kopplung PC 1715 mit dem KC 85/2 wurde zur Realisierung von drei Hauptanwendungsgebieten erarbeitet:

- Grafische Ausgabe von Kurven und Diagrammen im Zusammenhang mit wissenschaftlich-technischen Berechnungen
- Gewinnung und Darstellung von Flußschemata und Übersichtszeichnungen im Zusammenhang mit der rechnergestützten Projektierung von Chemieanlagen
- Aufbau eines vereinfachten Bildverarbeitungssystems (z. B. zur rechnergestützten Auswertung von Metalloberflächen).

Bild 4 zeigt ein Testbild für Grafikfunktionen, das mittels PC 1715 erstellt und auf dem KC 85/2 dargestellt wurde. Der Aufbau eines vereinfachten Bildverarbeitungssystems wurde realisiert, indem zusätzlich zur beschriebenen Hardwarekonfiguration ein Steuergerät zur Eingabe von Fernsehbilddaten entwickelt und über den zweiten PIO-Schaltkreis der Koppelkarte IHK/PC 2PIO an den PC 1715 angeschlossen wurde (Bild 5). Das entwickelte Eingabesteuergerät nutzt das Verfahren der spaltenweisen Abtastung des Fernsehsignals /5/. Es realisiert die Abtastung, Digitalisierung und Übertragung einer Bildpunktmatrix $P_{96,256}$ in den PC 1715. Die Digitalisierung erfolgt wahlweise in 6 bzw. 7 Bit Tiefe, das heißt, es ist eine Quantisierung in 64 bzw. 128 Graustufen möglich.

Die Bildpunktmatrix muß, um eine rechnergestützte Auswertung zu ermöglichen, oftmals in ein reines Binärbild (Schwarzweißdarstellung ohne Graustufen) transformiert werden. Die Wahl der exakten Triggerschwellen zur Erstellung des Binärbildes hängt von verschiedenen Faktoren, z. B. Qualität und Beleuchtung des Bildmaterials, eingestellte Blende der Fernsehkamera, ab. Bei Nutzung der Programme BILDOUT und BILDKC besteht für den Nutzer die Möglichkeit, durch die Darstellung aller 64 bzw. 128 Triggerschwellen bzw. Binärbildvarianten die effektivste binäre Darstellung auszuwählen. In den meisten Fällen kann durch die gezielte Auswahl von etwa 10 Bildern das optimale Binärbild ermittelt werden. Nach Ablauf der Bildtransformation erfolgt im PC 1715 die Bildverarbeitung. Zur Auswertung von Oberflächenstrukturen (z. B. von Schliffbildern, Bild 6) werden

- die rechnergestützte Korrektur des Kamerafehlers
- die Ermittlung des prozentualen Anteiles der Fehlstellen (schwarze Stellen) am Graubild
- die Klassifizierung der Fehlstellengröße in acht Klassen
- die Ermittlung der Fehlstellenverteilung auf dem Gesamtbild durchgeführt.

✉ KONTAKT ☎

Ingenieurschule Köthen, Sektion Anlagenbau, WB Physik. Bernburger Str. 52–57, Köthen, 4370; Tel. 6 73 25

Fortsetzung auf Seite 90

Das Softwarekonzept des KC 85/3

Dr. Werner Domschke
VEB Mikroelektronik „Wilhelm Pieck"
Mühlhausen

Computersoftware wird im allgemeinen in Basissoftware, dazu zählen u. a. Betriebssystem, Sprachübersetzer, Geräteanschlußprogramme, und Anwendersoftware eingeteilt.
Das Ziel des VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen besteht darin, den Computernutzern eine möglichst breite Palette von Basissoftware zur Verfügung zu stellen, um die Anwendungsbreite der Kleincomputer zu vergrößern. Da die Anwendersoftware stark vom jeweiligen Problem abhängt, wird darauf orientiert, daß die Programme vom jeweiligen Nutzer selbst erarbeitet werden. Ein Teil universell nutzbarer Anwendersoftware wird in Kürze angeboten werden. Deshalb soll im folgenden nur das Betriebssystem des KC 85/3 näher beschrieben werden; die andere verfügbare bzw. in Kürze erhältliche Basissoftware soll anderen Veröffentlichungen vorbehalten sein.

Der vorliegende Beitrag ist die Fortsetzung des Artikels „Kleincomputer KC 85/3", der in MP 2/87 ab Seite 56 erschienen ist.

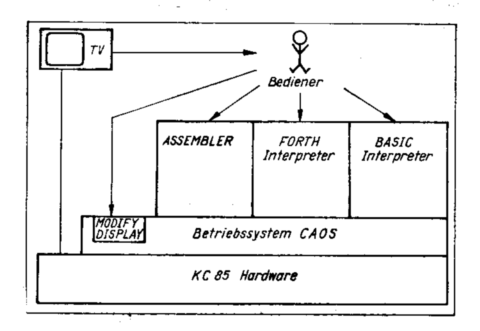


Bild 1 Die Stellung des Betriebssystems zum Bediener und zur Hardware

Das Betriebssystem HC-CAOS

Die Stellung des Betriebssystems in der anderen Basissoftware und in Bezug zum Bediener und der Computerperipherie zeigt Bild 1. Darin ist erkennbar, daß der Nutzer den Computer ohne weitere Basissoftware verwenden kann. Natürlich ist die Programmierung in Maschinensprache sehr umständlich. Deshalb ist der BASIC-Interpreter ebenfalls im ROM enthalten. Aber auch dabei stellt das Betriebssystem die zentrale Schnittstelle zwischen Bediener, BASIC-Interpreter und der Computerhardware dar. Das Betriebssystem HC-CAOS erfüllt u. a. folgende Aufgaben:

- Tastatureingabe
- Zeichendarstellung auf dem Bildschirm
- Punktsetzen und -löschen auf dem Bildschirm
- Farbeinstellung des Bildschirmes
- Fenstereinstellung auf dem Bildschirm
- Tonausgabe
- Datenein- und -ausgabe vom bzw. zum Kassettenrecorder
- Zeichenkettenumwandlung
- Schalten von Modulen und internen Speicherbereichen.

Bei der Konzipierung des Betriebssystems HC-CAOS wurde ebenso wie bei der Hardware Wert auf große Flexibilität und Anwenderfreundlichkeit bei voller Ausnutzung aller Hardware-Möglichkeiten gelegt. Deshalb konnte nicht von vorhandenen und bekannten Betriebssystemen, wie z. B. CP/M u. a., durch Abrüsten und Anpassen ausgegangen werden.

Die große Flexibilität von HC-CAOS wird durch die sogenannte Zeiger- und Tabellentechnik erreicht. Das Prinzip besteht darin, daß in festgelegten Speicherzellen im RAM die Anfangsadressen (Zeiger) von Tabellen und Unterprogrammen stehen. Durch Änderung des Zeigers kann der Anwender seine eigenen Tabellen und Unterprogramme nutzen bzw. die Systemfunktion nach seinen Bedürfnissen erweitern.

Das soll am Beispiel der Tastatureingabe erläutert werden. Jedem Signal, das der Tastaturprozessor sendet, ist über eine Tabelle ein interner Kode

zugeordnet. Dafür werden im KC 85/3 die ASCII-Zeichen verwendet. Die Umkodiertabelle steht im ROM und die Anfangsadressen dieser Tabelle im RAM.
Möchte der Anwender den Tasten der Tastatur andere Kodes zuordnen, so muß eine neue Kodetabelle programmiert und die Anfangsadresse dieser neuen Tabelle in die vorgesehenen RAM-Speicherzellen eingetragen werden. Dabei ist es belanglos, ob die neue Tabelle im RAM oder im ROM angeordnet ist. Damit lassen sich die Umlaute Ä, Ö, Ü, ä, ö, ü mit den Kodes 5BH, 5CH, 5DH, 7BH, 7CH, 7DH z. B. direkt von der Tastatur erzeugen. Voraussetzung für die Darstellung der Zeichen ist natürlich, daß ebenfalls die erforderlichen Zeichenbilder dieser Buchstaben im Computer enthalten sind. Auf diese beschriebene Weise lassen sich im HC-CAOS folgende Funktionen an die Anwenderbedürfnisse anpassen:

- Tastaturkodes
- Zeichenbilder
- System-Unterprogramme
- Steuerkodes des Bildschirmprogrammes
- Bildschirmausschnitte
- Console-Geräte (Standard: Tastatur und Bildschirm)
- jeweils zwei anwenderspezifische Ein- und Ausgabegeräte
- Pufferadresse für Ein-/Ausgabegeräte
- Reaktion auf die Hardcopy-Funktion
- Lage der System-Arbeitszellen im RAM
- Lage des Stack im RAM.

Das Prinzip bei der Änderung der Tastaturkodes ist bereits erläutert worden. Bei der Änderung der Zeichenbilder kann der Anwender, neben den 128 im ROM enthaltenen, weitere Zeichen definieren. Das geschieht auf die Weise, daß die in einem 8 × 8 Punktraster stehenden Zeichen entworfen und bitweise programmiert werden. Für ein Zeichen werden 8 Byte benötigt. Jeweils 64 Zeichen werden in einer Zeichenbildtabelle zu 512 Bytes zusammengefaßt. Im KC 85/3 können vier Zeichenbildtabellen gleichzeitig verwendet werden; mehrere können im Speicher enthalten sein, wenn bei Bedarf der entsprechende Zeiger geändert wird. Damit ist der Zeichenvorrat praktisch unbegrenzt.
Der Aufruf der System-Unterprogramme (UP) erfolgt im HC-CAOS über Programmverteiler, denen das gewünschte UP als Nummer übergeben wird. Die Anfangsadressen dieser UP sind in einer UP-Tabelle zusammengefaßt. Durch Kopieren dieser Tabelle in den RAM kann der Anwender die Zuordnung Nr. UP ändern oder diese Tabelle mit eigenen UP ergänzen.
Die Zuordnung Steuerkode-Reaktionsprogramm des Bildschirmtreibers erfolgt über die Steuerkode-Tabelle. Somit kann der Anwender für die vorhandenen Steuerzeichen seine eigenen Reaktionsprogramme definieren.
Bei der Festlegung von Bildschirmausschnitten muß der Anfang des Fensters (Zeile und Spalte) und die Fenstergröße definiert werden. Alle folgenden Bildschirmaktionen erfolgen nur innerhalb dieses definierten Fensters.
Im HC-CAOS sind für die Datenein- und -ausgabe standardmäßig drei Geräte vorgesehen: die Tastatur, das Fernsehgerät und der Kassettenrecorder. Die jeweiligen Treiberprogramme sind in die UP-Tabelle eingeordnet (UP-Nr. 00H, 01H, 04H und 05H). Für weitere zwei Ein- und zwei Ausgabegeräte sind in der UP-Tabelle Nummern reserviert. Die entsprechenden Treiberprogramme muß der Anwender selbst definieren bzw. werden für viele Standardgeräte vom Hersteller geliefert. Die Einbindung in das Betriebssystem erfolgt in der bereits beschriebenen Weise, daß die Anfangsadressen in dafür reservierte Speicherplätze eingetragen werden. Damit können Anwenderprogramme sehr universell und peripherieunabhängig geschrieben werden. Durch Laden spezieller Treiberprogramme kann nachträglich die gewünschte Gerätekonfiguration vom Anwender zusammengestellt werden.
Mit diesem Komfort ist es im HC-CAOS auch möglich, die Console-Geräte zu verändern. Durch Neudefinition des jeweiligen Zeigers kann die Systemeingabe z. B. vom UP-Nr. 04H (Tastatur) auf UP-Nr. 06H (Anwender-Eingabegerät 1) und die Systemausgabe von UP-Nr. 00H (Bildschirm) auf UP-Nr. 02H (Anwender-Ausgabegerät 1) umgeschaltet werden.
Diese beschriebene Zusammenarbeit des Betriebssystems HC-CAOS mit der Peripherie ist im Bild 2 dargestellt.
Eine weitere Besonderheit von HC-CAOS ist die Definition von Betriebssystemkommandos, die der Nutzer direkt ohne weitere Basissoftware ausführen kann. Diese Kommandos sind auch beliebig erweiterbar.
Jede CAOS-Funktion ist als Unterpro-

Literatur zum Artikel „KC 85/2 als intelligentes grafisches Display für den PC 1715"

/1/ Manual 1715. VEB Robotron Büromaschinenwerk Sömmerda
/2/ Kleincomputer KC 85/2 – Grundgerät. VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen
/3/ Servicehandbuch Robotron 1715. VEB Robotron-Büromaschinenwerk Sömmerda
/4/ Kleincomputer KC 85 – Beschreibung zum M006 BASIC. VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen
/5/ Vyhnal, D.: Entwicklung eines rechnergestützten Hochtemperaturthermographiesystems für den Meßbereich von 573 K bis 3280 K und seine Erprobung in der Schweißtechnik. Diss. IH Köthen 1984

Dr.-Ing. Werner Domschke (37) studierte von 1971 bis 1975 an der Technischen Universität Dresden in der Fachrichtung Technische Kybernetik und Automatisierungstechnik und promovierte dort 1982. Im VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen ist er an der Entwicklung des Kleincomputersystems KC 85 beteiligt und arbeitet als Abteilungsleiter Geräteentwicklung. Sein Spezialgebiet ist die Software des Kleincomputersystems.

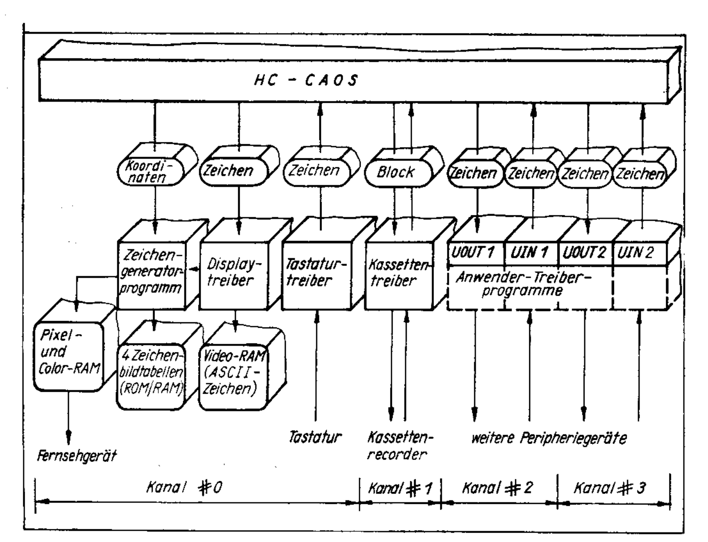


Bild 2 Datenaustausch von HC-CAOS mit der Peripherie

Tafel 1 Assemblerprotokoll für die Neudefinition von COPY

Adresse	Maschinencode	Anweisung	Bemerkung
0000	7F 7F	Vorsp: DEFW 7F7FH	;Prolog
0002	43 4F 50 59	DEFM 'COPY'	;Menüwort
0006	01	DEFB 1	;Epilog
0007	ED B0	LDIR	;Blocktransfer
0009	C9	RET	;Rücksprung zu ;CAOS

gramm definiert (mit RETURN abgeschlossen) und wird durch einen Vorspann gekennzeichnet. Der Vorspann besteht aus Prolog (2 Byte), normal 7F7FH, Menüwort (1 . . . 32 Zeichen: Ziffern, Großbuchstaben, Sonderzeichen,
Epilog (1 Byte) ØØH oder Ø1H.
Die Menüworte aller im aktiven Speicher stehenden CAOS-Funktionen werden nach RESET oder Aufruf der MENÜ-Funktion auf dem Bildschirm dargestellt. Durch Auswahl mit dem Cursor oder Neueingabe eines Menüwortes wird die jeweilige CAOS-Funktion gestartet. Die Anfangsadresse der Funktion ist dabei ohne Belang. Nach dem Menü der Funktion können noch bis zu 10 Argumente an die Funktion übergeben werden (Hexadezimalzahlen), die im RAM an definierter Stelle gespeichert sind. Die ersten drei Argumente werden vor dem Start der Funktion in die CPU-Register HL, DE und BC geladen und können sofort verwendet werden.
Dies soll an einem Beispiel erläutert werden.
Für das Kopieren von Speicherbereichen soll die CAOS-Funktion COPY neu definiert werden. Dazu sei das Assemblerprotokoll in Tafel 1 die Grundlage.

Bild 3 Steuerschleife des Betriebssystems HC-CAOS

Der Maschinenkode kann mittels der CAOS-Funktion MODIFY ab Adresse 0000 in den Speicher eingegeben und mittels der Funktion MENU auf dem Bildschirm angezeigt werden. Soll nun z. B. der Speicherbereich von 200H bis 500H (Länge: 300H) nach 1000H kopiert werden, so ist einzugeben

COPY 200 1000 300

Die Sprachübersetzer (z. B. Assembler, BASIC-Interpreter, FORTH-Interpreter usw.) und die Anwenderprogramme, die im Maschinenkode abgearbeitet werden, sind in diesem Sinn ebenfalls Erweiterungen der CAOS-Funktionen.
Beim KC 85/3 hat der Anwender weiterhin die Möglichkeit, unterschiedliche CAOS-Funktionen in Menü-Gruppen zusammenzufassen. Diese Menügruppen unterscheiden sich im Prolog. Das jeweils aktuelle Prolog-Byte wird in einer RAM-Zelle gespeichert. Nach dem Einschalten oder nach RESET ist 7FH als Prolog-Byte festgelegt. Somit können umfangreiche Anwenderprogramme, die als Maschinenkode definiert sind, in Hauptmenüs untergliedert werden. Damit ist eine anwenderfreundliche und übersichtliche Bedienerführung möglich.
Die Steuerschleife des Betriebssystems ist im Bild 3 wiedergegeben.
Abschließend sei darauf hingewiesen, daß die Entwicklung des Kleincomputersystems KC 85 aus Mühlhausen mit dem vorgestellten KC 85/3 nicht abgeschlossen ist. Das Betriebskollektiv wird weitere Zusatzmodule und Ergänzungsbaugruppen entwickeln und in die Produktion überleiten.

Literatur

/1/ WP DD 223 272 A1 G06 F 13/04
Verfahren und Anordnung zur digitalen Informationsaufzeichnung und -übertragung
/2/ WP DD 220 433 A1 G06 F 12/06
Anordnung zur Vergrößerung des adressierbaren Speicherbereiches für Mikrorechner

BASIC-Interpreter für KC 85/2 und KC 85/3

Karsten Schiwon, Sigrid Kollmeyer
VEB Mikroelektronik „Wilhelm Pieck"
Mühlhausen

Zum BASIC der Kleincomputer sind in den Artikeln /1/ und /2/ schon einige Ausführungen gemacht worden. Im folgenden Artikel sollen dem Anwender einige Unterschiede des RAM-residenten BASIC-Interpreters (Kassette C0111 BASIC-Interpreter) und des BASIC-Interpreters des Moduls M006 für den KC 85/2 sowie des im KC 85/3 enthaltenen BASIC-Interpreters aufgezeigt werden sowie Informationen, die ein besseres Verstehen des Interpreters ermöglichen, gegeben werden.
Um BASIC zu erlernen und spezielle Anwenderprogramme selbst zu erstellen, werden im Begleitbuch zur Kassette mit dem BASIC-Interpreter, in dem der Befehlssatz erläutert wird, Anleitungen gegeben. In den Begleitbüchern des Computers sind auch die Bedienung sowie Aufbau und Wirkungsweise des Systemprogramms beschrieben, deshalb soll hier nicht darauf eingegangen werden.

RAM- und ROM-BASIC

Mit dem Erwerb des Moduls M006 BASIC kann der Anwender des KC 85/2 seinen Computer auf die Softwaremöglichkeiten des KC 85/3 nachrüsten. Deshalb wird im folgenden vom ROM-BASIC, dem BASIC-Interpreter des Moduls bzw. des KC 85/3 und dem RAM-BASIC, BASIC-Interpreter der Kassette C0111 gesprochen.
Der erste wesentliche Unterschied der beiden Interpreter besteht darin, daß mit dem RAM-BASIC in der Grundvariante ein Arbeitsspeicher von etwa 5 KByte, mit der ROM-Variante etwa 15 KByte zur Verfügung stehen. Der RAM-Interpreter erlaubt es bei Einsatz der RAM-Erweiterungsmodule (zum Beispiel Modul M022 – EXPANDER RAM), einen Arbeitsspeicher von maximal etwa 21 KByte zu verwalten. Im Gegensatz dazu erlaubt die ROM-Variante, den Arbeitsspeicher des Interpreters bis auf etwa 47 KByte zu erhöhen. Die unterschiedliche Speicheraufteilung der minimalen und maximalen Ausbaustufen sind in dem Bild 1 dargestellt.
Der zweite Unterschied der BASIC-Varianten besteht im Befehlsumfang. Der ROM-BASIC-Interpreter besitzt einen erweiterten Befehlsumfang. Der Befehlsumfang des RAM-BASIC ist in Tafel 1 aufgeführt. Die neuen Befehle des ROM-BASIC sind in Tafel 2 dargestellt. Es sind alle BASIC-Programme, die auf dem KC 85/2 mit RAM-BASIC

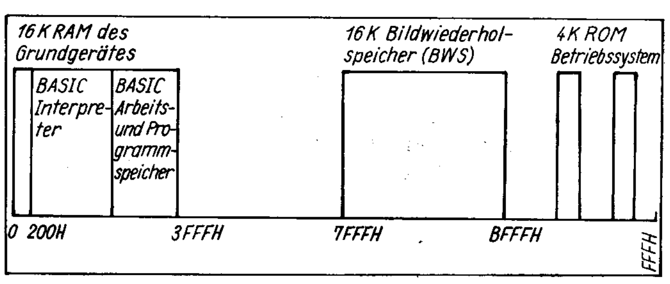


Bild 1a Minimale Variante mit RAM-BASIC

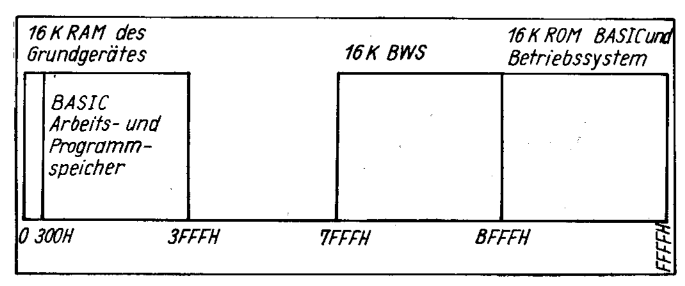


Bild 1b Minimale Variante mit ROM-BASIC

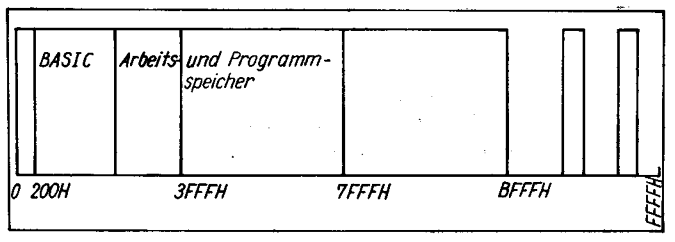


Bild 1c Maximalvariante RAM-BASIC

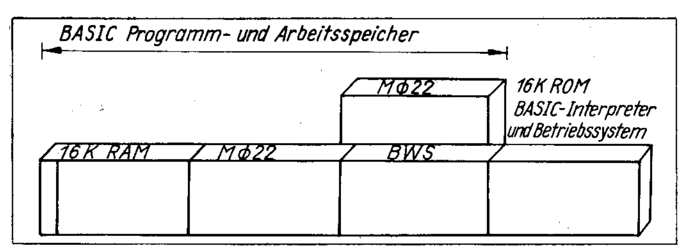


Bild 1d Maximalvariante ROM-BASIC

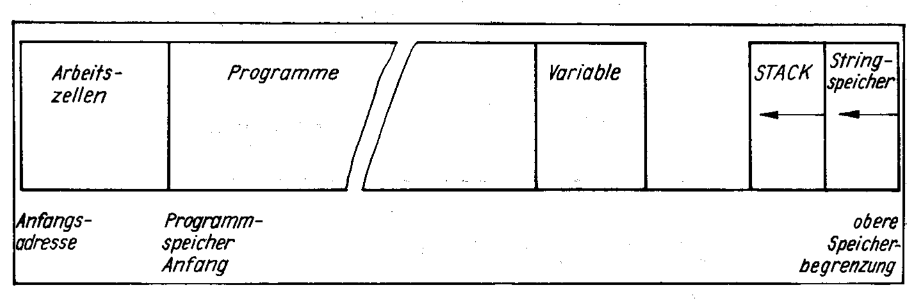


Bild 2 Aufteilung des dem BASIC-Interpreter zur Verfügung stehenden Speicherbereiches

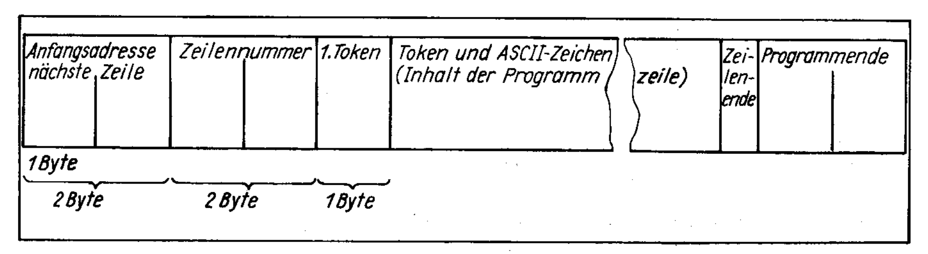


Bild 3 Aufbau einer BASIC-Programmzeile im Speicher

entwickelt wurden, auch ohne Umarbeitung auf dem ROM-BASIC lauffähig. Der volle Befehlsumfang des ROM-BASIC wird in dem Begleitbuch zum KC 85/3 und zum Modul M006 beschrieben. In Tafel 3 ist eine vollständige Übersicht über die Token des Interpreters gegeben.

Unterschiede im BASIC-Befehlssatz KC 85/1 und KC 85/3

Im wesentlichen stimmen beide Befehlssätze überein, es wird der gleiche Interpreterkern benutzt. Dieser Kern wurde hardwarespezifisch für die Computer erweitert, wodurch Unterschiede bedingt sind.
Die Anwendung des KC-85/1-Kommandos BORDER führt beim KC 85/3 zu keiner Reaktion. Das Kommando JOYST führt zu einem Sprung auf die Adresse 1FDH, wo ein Sprung zu einer vom Anwender erstellten Routine eingetragen werden muß.
Im allgemeinen können Programme des KC 85/1 für den KC 85/3 bzw. Programme des KC 85/3 für den KC 85/1 angepaßt werden. Dies gilt nicht, wenn im Programm auf die Systemzellen zugegriffen wird. Das betrifft die Kommandos POKE, DOKE, INP, OUT, VPEEK, PEEK, DEEK. Programme, die diese Kommandos enthalten, können nur von Anwendern angepaßt werden, die beide Systeme genau kennen.
Der komplette Quasi-Grafik-Satz des KC 85/1 kann beim KC 85/2 oder KC 85/3 mit der Kassette C0163 Zeichenbilder geladen und über das BASIC-Kommando PRINT CHR$ (n) mit $128 \leq n \leq 255$ angezeigt werden.
Eine Anpassung von Programmen, in denen Steuerhebel genutzt werden, ist z. Z. nicht möglich. Um die Steuerhebel des KC 85/1 an den KC 85/3 anschließen zu können, ist ein Adapter für den in Entwicklung befindlichen digitalen E/A-Modul M001 notwendig. Dieser Modul ist geeignet, zwei Steuerknüppel abzufragen. Die Farbkommandos haben beim KC 85/1 nur Wirkung, wenn die zusätzliche Farbausrüstung vorhanden ist.
Dabei ist zu beachten, daß die Farbkodierungen unterschiedlich sind. Beim Befehl WINDOW ist zu beachten, daß die maximal mögliche Anzahl von Zeilen unterschiedlich ist (KC 85/1 hat 24 Zeilen; KC 85/2 hat 32 Zeilen).

Speicheraufbau

Der interne Aufbau des Programm- und Arbeitsspeicherbereiches der beiden BASIC-Interpreter (RAM- und ROM-Variante) ist gleich. Wie aus den Bildern 1a bis 1d zu entnehmen ist, haben die beiden BASIC-Varianten unterschiedliche Anfangsadressen der Speicherbereiche für ihre Arbeitszellen, Programme und Variablen. Der Speicherbereich, der vom BASIC belegt wird, kann auch mehr oder minder groß sein. Mit dem Kaltstart des Interpreters, Menü-Anweisung „BASIC", werden die Systemzellen des Interpreters in den Grundzustand gesetzt. Dabei wird auch, nachdem die Abfrage „MEMORY END :" vom Bediener beantwortet wurde, die Speicherobergrenze festgelegt. Deshalb zeigen die Bilder 1a bis 1d auch den jeweils größtmöglichen Speicherbereich der einzelnen Varianten für BASIC-Programme (ohne Begrenzung des Programm- und Arbeitsspeicherbereiches unter die mögliche obere Grenze).
Im Bild 2 ist die relative Einteilung des Speicherbereiches, der dem BASIC-Interpreter zur Verfügung steht, bezüglich

Tafel 1 Befehlsumfang des RAM-BASIC – das in /2/ beschriebene BASIC wurde für den KC 85/2 um die aufgeführten Befehle erweitert

BLOAD
Rufen des Betriebssystemprogramms LOAD zum Einlesen von Maschinenprogrammen während der Arbeit des BASIC-Interpreters

COLOR V, H
Einstellen der Vorder- und Hintergrundfarbe,
$0 \leq V \leq 15$

LOCATE Z, S
Positionieren des Cursors an einer bestimmten Stelle innerhalb des Fensters, Z = Zeile im Fenster, S = Spalte im Fenster

PSET X, Y, [,F]
Setzen des Punktes (X, Y) in der Farbe F,
$0 \leq X \leq 319$, $0 \leq Y \leq 255$, $0 \leq F \leq 31$, die Angabe der Farbe kann entfallen

PRESET X, Y
Löschen des Punktes (X, Y)

SOUND Z1, V1, Z2, V2, LS, TD
Ausgabe von Tönen mit steuerbarer Tonhöhe, Tonlänge und Lautstärke

Z1 – Zeitkonstante für CTC Kanal 1
V1 – Vorteiler für CTC Kanal 1
Z2 – Zeitkonstante für CTC Kanal 2
V2 – Vorteiler für CTC Kanal 2
LS – Lautstärke
TD – Tondauer
$1 \leq Z \leq 255$ (Z = 0 Ton ausgeschaltet)

$$V = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

$1 \leq LS \leq 31$ (LS = 0 Tonausgabe über TV gesperrt)
$1 \leq TD \leq 255$ (TD = 0 Dauerton)

VPEEK (AD)
Lesen des Inhaltes der Speicherzelle AD im Bildwiederholspeicher

VPOKE AD, W
Beschreiben der Speicherzelle AD im Bildwiederholspeicher mit dem Wert W

Tafel 2 Befehle, die nur im ROM-BASIC enthalten sind

CIRCLE XM, YM, R [,F]
Zeichnen eines Kreises mit Mittelpunkt XM, YM und Radius R in der Farbe F; die Angabe der Farbe kann entfallen

CLOSE r # n
Schließen eines Kanals n

$$r = \begin{Bmatrix} O \\ I \end{Bmatrix} \begin{matrix} \text{Ausgabe} \\ \text{Eingabe} \end{matrix}$$

JOYST (n)
Vorbereitete Funktion für Steuerknüppelabfrage, bei Aufruf erfolgt Sprung zur Adresse 1FDH

KEY N
Belegen der Funktionstasten
$1 \leq N \leq 12$

KEYLIST
Anzeige der Funktionstastenbelegung

LINE XA, YA, XE, YE [,F]
Ausgabe einer Linie vom Punkt (XA, YA) zum Punkt (XE, YE) in der Farbe F, die Angabe der Farbe kann entfallen,
$0 \leq XA, XE \leq 319$
$0 \leq YA, YE \leq 255$

OPEN r # n „NAME"
Eröffnen einer Kanaloperation mit Namensübergabe, gleich Typ wie LIST ≠ 1

RANDOMIZE
Initialisieren des Zufallsgenerators

SWITCH M, K
Ausgabe des Steuerbytes M an das Modul auf der Adresse K

INSTR (A$, B$)
Ermitteln der Position, ab welcher A$ in B$ enthalten ist; ist A$ erst unvollständig und danach vollständig in B$, wird der vollständige String nicht erkannt

VGET$
Ergibt den Inhalt der Cursorposition als String

CSRLIN (N)
Liefert die Nummer der Zeile, in welcher der Cursor steht,
N = 0 bezüglich Bildschirm
N ≠ 0 bezüglich WINDOW

PTEST (X)
Testet, ob Bildpunkt X gesetzt ist,
Ergebnis: 0 Punkt war nicht gesetzt
1 Punkt war gesetzt

Tafel 3 Token des KC 85/2 und /3 hexadezimal geordnet

	8	9	A	B	C	D	E	F
0	END	OUT	LIST	∧	EXP	LOAD	PAPER	RANDOMIZE
1	FOR	ON	CLEAR	AND	COS	TRON	AT	VGET$
2	NEXT	NULL	CLOAD	OR	SIN	TROFF	COLOR	LINE
3	DATA	WAIT	CSAVE	>	TAN	EDIT	SOUND	CIRCLE
4	INPUT	DEF	NEW	=	ATN	ELSE	PSET	CSRLIN
5	DIM	POKE	TAB	<	PEEK	INKEY$	PRESET	
6	READ	DOKE	TO	SGN	DEEK	JOYST	BLOAD	
7	LET	AUTO	FN	INT	PI	STRING$	VPEEK	
8	GOTO	LINES	SPACE	ABS	LEN	INSTR	VPOKE	
9	RUN	CLS	THEN	URS	STR$	RENUMBER	LOCATE	
A	IF	WIDTH	NOT	FRE	VAL	DELETE	KEYLIST	
B	RESTORE	BYE	STEP	INP	ASC	PAUSE	KEY	
C	GOSUB	!	+	POS	CHR$	BEEP	SWITCH	
D	RETURN	CALL	–	SQR	LEFT$	WINDOW	PTEST	
E	REM	PRINT	*	RND	RIGHT$	BORDER	CLOSE	
F	STOP	CONT	/	LN	MID$	INK	OPEN	

Anfangsadresse und Speicherobergrenze, angegeben. Die Anfangsadresse dieser Speicherbereiche ist für RAM-BASIC die 2A00H, für ROM-BASIC die 300H. Der Programmspeicher des RAM-BASIC fängt bei der Adresse 2B01H, der des ROM-BASIC bei 401H, an. An dieser Stelle sei noch einmal vermerkt, daß jede Zahl im BASIC mit insgesamt 4 Byte dargestellt wird, womit die Rechengenauigkeit des Interpreters festgelegt ist. Nähere Rechengenauigkeiten können erreicht werden, indem eine eigene Arithmetik in Form von Maschinenprogrammen geschrieben wird oder eine sogenannte Stringarithmetik im BASIC-Programm realisiert wird. Der Programmspeicherbereich des Interpreters enthält das BASIC-Programm, das sich aus den nacheinander angeordneten Programmzeilen, die nicht mehr als 72 Zeichen enthalten, zusammensetzt. Der Aufbau einer Programmzeile ist in Bild 3 dargestellt. Die einzelnen Programmzeilen sind durch 1 Trennbyte (00) getrennt. Der Stringspeicherbereich kann mit dem Befehl CLEAR verändert werden. Ein String ist nicht länger als 255 Byte (ASCII-Zeichen). Für Felder, die mehr als 10 Elemente enthalten sollen, muß eine Dimensionierung erfolgen, damit der Interpreter die Daten der Reihenfolge nach Ablegen kann.
Die Dimensionierung kann innerhalb eines BASIC-Programms nicht ohne programmtechnische Bearbeitung nur durch den DIM-Befehl verändert werden. Sollen Felder umbenannt werden und andere Dimensionierung erhalten, kann das durch Umspeichern (Duplizieren) der Elemente geschehen.

Literatur

/1/ Riedner, Dämne: Software für Heimcomputer Z 9001. rfe Berlin 33 (1984) 9
/2/ Keller, G.: BASIC für Heimcomputer Z 9001, rfe Berlin 33 (1984) 9

WordPro '86 für KC 85/2 und /3 auf Kassette verfügbar

Dieses 80-Zeichen-Textverarbeitungssystem für die genannten Kleincomputer wurde bereits beschrieben (Funkamateur 7 und 8/86, rfe 8 und 10/86). Auf Veranlassung des VEB Robotron-Vertrieb Berlin erhielt es eine auf alle nötigen Baudraten hin modifizierbare V.24-Druckerausgaberoutine. Mit dieser Ergänzung wurde es im Zusammenspiel mit dem Matrixdrucker K 6313 mit V.24-Port erfolgreich getestet.
Für Drucker mit Parallelschnittstelle kann jedoch auch die vorher benutzte Centronics-Routine eingebaut werden. Das ist besonders wichtig für jene KC 85/2 aus der ersten Serie, in denen der bekanntlich erst später entwickelte V.24-Modul nicht arbeitet. Das 8-K-Maschinenprogramm WordPro '86 verwaltet im Grundgerät 109 und mit 16-K-Modul 314 Zeilen zu je maximal 80 Zeichen. Es paßt sich der jeweiligen Konfiguration automatisch an. Schriftdarstellung ist in schwarz auf weiß und umgekehrt wählbar.
Anfragen zum Bezug der Kassette sind zu richten an den VEB Robotron-Vertrieb Berlin, Abt. Vertrieb Software und Dokumention, Postfach 1235, 1986 Berlin.

Arbeit mit BASIC-Datenfeldern beim KC 85/3

Werden in BASIC-Programmen größere Datenmengen verarbeitet, wie z. B. in Namensverzeichnissen, ist es günstig, diese Daten in Feldern getrennt vom Programm abzuspeichern. In manchen Fällen ist es aber notwendig, mehrere verschiedene Datenfelder gleicher Struktur, je nach aktueller Problemstellung, zu verarbeiten.
Ein Anwendungsfall ist z. B. ein Personalprogramm, welches aus Speicherplatzgründen in mehrere Datenfelder für unterschiedliche betriebliche Strukturen unterteilt ist. Da es beim BASIC-Interpreter der KC-Serie nicht möglich ist, den Namen der Kassettenaufzeichnung als Stringvariable anzugeben, muß der Name direkt im Programm hinter der Ladeanweisung in Anführungszeichen stehen. Somit muß vor jedem Laden eines Feldes mit einem anderen Namen die entsprechende Programmzeile mit der 'EDIT'-Anweisung geändert werden. Als Ausweg bietet sich hier an, den Namen als Stringvariable einzugeben und die einzelnen Zeichen in die entsprechende Stelle direkt in das Programm „hineinzupoken". Tafel 1 zeigt das zugehörige Teilprogramm. Das Einlesen des Datenfeldes erfolgt durch Aufruf des Unterprogrammes auf Zeile 20 von beliebiger Stelle des Programmes. Die Zieladressen für den Namen lassen sich einfach bestimmen durch Verlassen des BASIC-Interpreters nach Eingabe des Programmes aus Tafel 1. Im Betriebssystem erfolgt die Anzeige des HEX-Dumps mit der DISPLAY-Anweisung ab Adresse 401H (DISPLAY 401 440 'ENTER'). Werden die Programmzeilen des angegebenen Beispiels direkt übernommen, kann die Adreßbestimmung entfallen. Eine einfache Kontrollmöglichkeit ist das Abarbeiten und anschließende Auflisten des Programmteils. Der neue Name muß dann an der richtigen Stelle im Programm eingetragen sein. Der Name muß acht Zeichen lang sein. Das vorgestellte Verfahren kann für den KC 85/3 und für den BASIC-Modul M006 beim KC 85/2 verwendet werden, ebenso für den BASIC-Modul des KC 85/1.
Ein weiteres Problem ist das Suchen nach Dateien auf der Kassette. Findet der BASIC-Interpreter einen 'falschen' Namen, so meldet er sofort „IO ERROR", und das Programm wird abgebrochen. Tafel 2 zeigt ein Teilprogramm, mit dem es möglich ist, in einer vorgebbaren Anzahl von Dateien nach einer bestimmten zu suchen und diese zu laden. Die Namenseingabe erfolgt, wie bereits vorgestellt. Eine Programmfortsetzung nach Fehlerabbruch ist über 'GOTO'-Anweisung möglich. Um diese nicht von Hand eingeben zu müssen, kann das Systemunterprogramm benutzt werden, welches die Funktionstasten realisiert. In den Speicherzellen B7D1H und B7D2H steht ein Zeiger auf einer Zeichenkette, die bei aktivierter F-Taste ausgegeben wird. Die Tastatureingabe wird auf die Zeichenkette umgeschaltet, wenn das Bit 6 der Speicherzelle 1F8H gesetzt ist. Die Zeichenkette wird mit einem Byte 00H abgeschlossen. In Programmzeile 130 steht die Zeichenkette, die in der folgenden Programmzeile in den freien Speicher ab Adresse 01H „gepokt" wird. In Zeile 180 wird sie aktiviert. Wird die richtige Datei geladen, so erfolgt in Zeile 300 ein Zurückstellen auf normale Tastatureingabe für die weitere Programmabarbeitung. Das hier vorgestellte Verfahren zum Umgehen einer Tastaturanforderung ist nicht nur auf eine einzelne Anweisung beschränkt, es kann auch in anderen Fällen genutzt werden, wenn der BASIC-Interpreter in die Kommandoeingabe übergeht (z. B. nach dem Programmladen oder -retten). Hier ergibt sich für den Experimentierfreudigen noch ein weites Feld.
Die Zeilen 110 bis 120 sind notwendig, damit das Programm auch bei auf 48 KByte ausgebauten Speichern funktioniert. Hierbei wird beim Aufruf der „IO ERROR"-Routine (auf Adresse DC82H) durch Umstellen des Programmzeigers im RAM (B7C9H) der zusätzliche Aufruf der Systemroutine zum Abschalten des Bildwiederholspeichers realisiert. Diese drei Programmzeilen sollten am Anfang jedes BASIC-Programmes stehen, welches bei voll ausgebautem Speicher (zwei 16-KByte- oder ein 64-KByte-RAM-Modul) mit Kassettendateien arbeitet.

```
 10 GOTO30
 20 CLOAD*"AAAAAAAA";A$:RETURN
 30 DIMA$(100):INPUT"NAME:";N$
 40 FORZ=0TO7:POKE1040+Z,ASC(MID$(N$,Z+1)):NEXT
```

Tafel1 Teilprogramm zur Namenseingabe

```
 10 GOTO100
 20 CLOAD*"12345678";FD$
 30 PRINT"Datei gefunden":GOTO300
100 DATA205,27,240,195,130,220
110 FORZ=0TO5:READA:POKEZ,A:NEXT
120 VPOKE14281,0:VPOKE14282,0
130 A$="GOTO200"+CHR$(13)+CHR$(0):DIMFD$(30)
140 Q=LEN(A$):FORZ=1TOQ:POKEZ+7,ASC(MID$(A$,Z,1)):NEXT
150 CLS:INPUT"Anzahl der zudurchsuchenden Dateien :";X
160 INPUT "Name der gesuchten Datei :";X$:X$=X$+STRING$(8," ")
170 FOR Z=1TO8:POKE1040+Z,ASC(MID$(X$,Z,1)):NEXT
180 POKE504,PEEK(504)OR64:VPOKE14289,8:VPOKE14290,0
190 GOTO20
200 X=X-1:IFX>0THEN180
210 PRINT"Datei nicht gefunden":END
300 POKE504,PEEK(504)AND191
310 !PROGRAMMANFANG
```

Tafel2 Programmbeispiel für Dateisuche

K.-D. Kirves

Schnittstellen für Kleincomputer

An der Friedrich-Schiller-Universtität wurde im Februar 1986 ein Technikum für Laborautomatisierungs- und -rationalisierungsmittel (LAURA) ins Leben gerufen. Die Mitarbeiter dieses Technikums befassen sich mit dem Einsatz der Mikrorechentechnik zur Rationalisierung der wissenschaftlichen Arbeit an Universitäten und Hochschulen und zur Rationalisierung des Forschungsprozesses.
Für die Kleincomputer KC 85/1, KC 85/2 und KC 85/3 wurden folgende Schnittstellen bzw. Koppelmoduln am Technikum entwickelt:

• *IFSS- und V.24-Modul*

Diese Interfaces enthalten einen IFSS (20-mA-Stromschleife) oder V.24-Datenübertragungskanal, der den Anschluß von Ein-/Ausgabegeräten mit serieller Informationsübertragung gestattet.
Betriebsweise: asynchron, duplex oder halbduplex
Format: 5 ... 8 Datenbits
Parität: ohne, gerade oder ungerade
Übertragungsrate: 100, 150, 200, 300, 600, 1200, 2400, 4800 oder 9600 Baud

• *GNI-Modul*

Die Baugruppe enthält einen GNI-Übertragungskanal, der den Anschluß von Ein-/Ausgabegeräten mit paralleler Informationsübertragung über eine PIO-Datenschnittstelle gestattet. Durch die wechselseitige Bereitschaftsabfrage wird eine maximale Übertragungsrate und Sicherheit erreicht.
Die Baugruppe wurde vor allem entwikkelt zum Anschluß der drei Varianten des Seriendruckers SD 1157, sind aber auch für alle anderen E/A-Geräte benutzbar, die über diese standardisierten Schnittstellen verfügen.

• *Anschluß der Schreibmaschinen S 6005 und S 6011*

Um die Schreibmaschinen als preiswerte Ein-/Ausgabegeräte an den Kleincom-

putern nutzen zu können, wurden entsprechende Anschlußsteuerungen entwickelt, die nachträglich in die Schreibmaschine eingesetzt werden können. Drei verschiedene Betriebsarten sind möglich:

- unabhängiger Betrieb von Computer und Schreibmaschine
- Schreibmaschine als Ausgabegerät
- Schreibmaschine als Eingabemedium.

Für sämtliche Baugruppen existiert eine Treibersoftware, die die Einbindung ins Betriebssystem beinhaltet, so daß z. B. über BASIC oder Editor/Assembler ein- bzw. ausgegeben werden kann. Für den Schreibmaschinenanschluß steht zusätzlich noch ein Textverarbeitungssystem zur Verfügung.

• *Koppelmodul KC 85/2 und KC 85/3 mit PC 1715*

Diese Baugruppe gestattet den Datenaustausch zwischen dem Personalcomputer PC 1715 und dem Kleincomputer. Über diese Kopplung kann somit z. B. der Kleincomputer als Vollgrafik-Einheit am PC verwendet werden.
Weitere Programmpakete befinden sich in Entwicklung.

Prozeßkoppelmodul

Die programmierbare Kleinsteuerung PKS 8 dient zur Ankopplung eines Prozesses an einen übergeordneten Steuerrechner. Durch die Anwendung eines Einchip-Mikrorechners besitzt die Baugruppe eine eigene Intelligenz.

Parameter

prozeßseitig:
3 × 8 Bit digitale Ausgabe, statisch
2 × 8 Bit digitale Eingabe
1 × Analogeingabe 0 . . . 10 V, einstellbar
rechnerseitig:
standardisierte serielle Schnittstelle IFSS oder V.24, umschaltbar
Softwareseitig stehen das Betriebssystem für die PKS 8 sowie Kopplungsprogramme für K-1520-Rechner (z. B. BC A5120, BC A5130, PC 1715, MC 80) zu Verfügung.
Die Nachnutzung beinhaltet: Dokumentation, unbestückte Leiterplatte, u. U. geprüfte Baugruppe und entsprechende Software.

Anfragen sind zu richten an: Friedrich-Schiller-Universität Jena, Sektion Technologie für den WGB, Technikum LAURA, Ernst-Thälmann-Ring 32, Jena, 6900, Tel.: 82 24 3 45.
Prof. Dr. G. Entreß
M. Günzel, M. Reinhardt

Zur Diskussion gestellt:

Tastaturorientierter Rechnerdialog – ein Ausblick

Jürgen Schlenzig
Meiningen

Es besteht weitgehend Einigkeit darüber, daß die Tastatur für absehbare Zeit Haupteingabemittel für Computer bleiben wird. Daraus erwächst die Forderung nach Verbesserung der Arbeit mit der Tastatur.
Es ist möglich und sinnvoll, mit den Mitteln der Mikrocomputertechnik die Tastatur zu einem dem Nutzer entgegenkommenden Werkzeug umzugestalten.
Dies führt zu einer höheren Nutzerfreundlichkeit, eröffnet neue anwendungstechnische Möglichkeiten und ersetzt Mechanik durch Elektronik.
Vorgeschlagen als Basis einer Tastaturweiterentwicklung wird das in /1/ und /2/ beschriebene Prinzip einer aktiven Tastatur.
Diese besteht aus 12 neutralen Eingabetasten, einer SPACE-Taste (Zwischenraum) sowie den Funktionstasten NEXT, OC (other code, Funktionsumschaltung Buchstaben/Ziffern/Steuerzeichen) und UC/LC-Umschaltung.
Eine zugeordnete Anzeigeeinheit, die durch den Tastaturtreiber gesteuert wird, gestattet ein Arbeiten mit dem vollen Zeichenumfang alphanumerischer Tastaturen.
Der ergonomische Vorteil besteht darin, daß die Tastatur bei Texteingabe aufgrund eines sprachstatistisch begründeten Algorithmus dem Nutzer entgegenkommt.
Wird zur Anzeige der dynamischen Tastaturbelegung ein Teil des Bildschirmes verwendet, ergibt sich eine einfache Realisierungsmöglichkeit für Eigenbautastaturen:
Ein Eingabeblock von 12 plus 14 Tasten wird als 4 × 4-Matrix über ein PIO-Tor angeschlossen. Zusätzlich zum Einbinden der Tastaturroutine in den Monitor werden 30 Tastaturbelegungsmuster zu je 12 Zeichen auf EPROM gespeichert.
Die Lösung ist hinsichtlich Material-, Kosten- und Platzbedarf bei voller Tauglichkeit für alphanumerische Eingabe nicht zu unterbieten.
Es sei betont, daß die dynamische Tastaturbelegung für den Nutzer vorteilhaft ist, also keinesfalls eine Behelfslösung darstellt.
Mustererprobungen ergaben eine überraschend schnelle Gewöhnung an die wechselnde Tastaturbelegung. Blindschreiben einzelner Wortteile oder Worte war nach kurzer Zeit möglich.
Mit höherem Aufwand, aber gleichem Arbeitsprinzip ist unter Einbeziehung von Prozessor und EPROM-Speicher (besser: U881), sowie einer alphanumerischen LED- oder LCD-Anzeige eine zu Normaltastaturen stecker- und softwarekompatible Aktivtastatur realisierbar.

Eine anwendungstechnische Erweiterung bietet sich, wenn die Aktivtastatur über ein Anwenderprogramm beeinflußbar wird. So könnte bei Datenerfassungsterminals in Abhängigkeit von der Eingabemaske numerische oder Alpha-Eingabe erzwungen werden, indem das Programm die ansonsten mit der OC-Taste vorzunehmende Umschaltung zwischen Buchstaben, Ziffern und Sonderzeichen bewirkt. Dies setzt einen realisierten Tastaturausgabekanal voraus. Der Umfang von Datenprüfroutinen kann reduziert werden, da bestimmte Fehlerklassen nicht mehr auftreten.
Der Weg ist nun nicht mehr weit zur intelligenten Tastatur: Sie beinhaltet einen Wortspeicher, den das Tastatursteuerprogramm verwaltet und der nach Eingabe weniger signifikanter Buchstaben dem Bediener wahrscheinliche Komplettierung vorschlägt (auf Tastaturanzeige oder Bildschirm).
Das Betätigen einer Bestätigungstaste führt dann zur Übernahme kompletter Worte; andernfalls wird bei abweichender weiterer Eingabe ein neuer Vorschlag gebracht. Dieses Eingabeverfahren ist an keine spezielle Tastaturbauform gebunden. Es ist damit eine Tastatur realisiert, die mehr Zeichen übertragen kann, als Tasten betätigt wurden.

Dem Argument, daß derartige Lösungen einer perfekten Schreibkraft keinen Vorteil bringen, kann nicht gefolgt werden: So wie die Lage der Tasten auf einer Schreibmaschinentastatur gelernt wird, so ist auch die wechselnde Belegungsvorgabe bei einer aktiven Tastatur oder die Wortvorgabe bei wortspeichergestützter Eingabe mittels intelligenter Tastatur erlernbar.

Literatur

/1/ Schlenzig, J.: Die aktive Tastatur. rfe Berlin 34 (1985) 10, S. 660
/2/ Wirtschaftspatent DD 234 953 A1
/3/ Elektronik-Rechtschreibung, rd Berlin 23 (1986) 1, S. 3

Claßen, L.
P8000 – ein universelles 16-Bit-Mikrorechnerentwicklungssystem
Mikroprozessortechnik, Berlin 1 (1987) 3, S. 68
Der Beitrag behandelt umfassend ein universelles Programmier- und Entwicklungssystem für die 8- und 16-Bit-Mikroprozessorfamilien U8001/U8002, K1810 WM86, U880 und U881/U882, welches Leistungscharakteristiken moderner Arbeitsplatzcomputersysteme mit Multi-User-/Multi-Task-Eigenschaften aufweist. Zahlreiche Softwarewerkzeuge der 16-Bit-Mikrorechnerklasse in einer UNIX-kompatiblen Programmierumgebung, ergänzt durch alle bewährten 8-Bit-Softwaresysteme, werden zur Verfügung gestellt.

Claßen, L.
P8000 – универсальная 16-Иразрядная система проектирования микро-ЭВМ
Mikroprozessortechnik, Berlin 1 (1987) 3, стр. 68
В статье широко рассмотрена универсальная система программирования и проектирования для 8-миразрядных и 16-тиразрядных микропроцессорных семейств U8001/U8002, K1810 WM86, U880 и U881/U882, которая имеет рабочие характеристики современных вычислительных систем рабочих мест с множественным доступом и мультизадачной обработкой. Предоставлены в распоряжение многочисленные програмные средства класса 16-тиразрядной микро-ЭВМ в среде программирования, совместимой с UNIX, дополнены всеми испытанными 8-миразрядными системами программного обеспечения.

Claßen, L.
P8000 – a Universal 16-Bits Microcomputer Development System
Mikroprozessortechnik, Berlin 1 (1987) 3, pp. 68
For the 8- and 16-bits microprocessor families U8001/U8002, K1810 WM86, U880, and U881/U882 the author treats comprehensively a universal programming and development system showing the performance features of modern working station computer systems with multi-user and multi-tasking characteristics. The system offers numerous software tools of the 16-bits microcomputers in a UNIX-compatible programming environment supplemented by all the proved 8-bits software systems.

Heuer, H.
Parallelverarbeitende Rechnersysteme
Mikroprozessortechnik, Berlin 1 (1987) 3, S. 71
Ein spürbarer Zuwachs der Verarbeitungsgeschwindigkeit von Rechnersystemen kann künftig nur auf der Basis innovativer Rechnerarchitekturen erzielt werden. Der Autor untersucht den gegenwärtigen Stand und stellt zur Demonstration der Leistungsfähigkeit der Parallelverarbeitung ein universelles und flexibles Multiprozessorsystem mit mehrfachem Befehls- und Datenstrom vor.

Heuer, H.
Параллельно обрабатывающие вычислительные системы
Mikroprozessortechnik, Berlin 1 (1987) 3, стр. 71
Значительное приращение скорости обработки вычислительных систем в будущем возможно достигать только на основе новых вычислительных архитектур. Автор проверяет настоящее состояние и представляет для демонстрирования производительности параллельной обработки универсальную и гибкую мультипроцессорную систему с многократным потоком команд и данных.

Heuer, H.
Computer Systems with Parallel Processing
Mikroprozessortechnik, Berlin 1 (1987) 3, pp. 71
In future a considerable increase of the computer systems processing speed will only be gained on the basis of innovative computer architectures. The author studies the state of the art and presents a universal and flexible multiprocessor system with multiple command and data streams in order to demonstrate the efficiency of parallel processing.

Heymer, V.
Lokale Rechnernetze mit OSI-Architektur
Mikroprozessortechnik, Berlin 1 (1987) 3, S. 74
Um eine effektive Zusammenarbeit von Gerätesystemen unterschiedlicher Hersteller in lokalen Netzen zu ermöglichen, wurde begonnen, national und international an einer Standardisierung zu arbeiten. Dabei setzt sich gegenwärtig das von der ISO standardisierte OSI-Referenzmodell zur Verkopplung offener Systeme als Basis auch für lokale Netze durch. Forschungs- und Entwicklungsarbeiten für OSI-gerechte lokale Netze werden im Hochschulwesen, in der AdW und im Kombinat Robotron durchgeführt.

Heymer, V.
Локальные вычислительные сети с архитектурой ОСИ
Mikroprozessortechnik, Berlin 1 (1987) 3, стр. 74
С целью обеспечения эффективного сотрудничества приборных систем различных изготовителей в локальных сетях начата работа над стандартизацией в национальных и международных рамках. При этом в настоящее время имеет успех стандартизованная ИСО опорная модель ОСИ для связи разомкнутых систем как основа также и для локальных сетей. Исследовательские работы для локальных сетей и ОСИ проводят в вузах, академии наук и комбинате РОБОТРОН.

Heymer, V.
Local Computer Networks with OSI-Architecture
Mikroprozessortechnik, Berlin 1 (1987) 3, pp. 74
In order to enable an efficient co-operation of different producers' apparatus systems via local networks a standardized by ISO for coupling open systems is being adopted as a basis fo local networks, too. Research and development work for OSI-oriented local networks are carried out by colleges, academy of sciences and Kombinat Robotron.

Schmidt, J.
Testschaltkreis AD31
Mikroprozessortechnik 1 (1987) 3, S. 76
Es werden das Schaltungsprinzip, die wichtigsten internen Funktionsgruppen und die Funktionsweise eines monolithisch integrierten 8-Bit-A/D-Umsetzers nach dem sukzessiven Approximationsverfahren in MOS-Technologie erläutert. Der Baustein ist für die Erfassung mittelschneller Echtzeitprozesse, wie z. B. akustische Signalerfassung, geeignet.

Schmidt, J.
Преключательная схема контроля АД31
Mikroprozessortechnik, Berlin 1 (1987) 3, стр. 76
Пояснены принцип схем, основные внутренние функциональные группы, и принцип действия монолитически интегрированного восьмиразрядного аналого-цифрового преобразователя по принципу последовательного приближения в технологии МОП. Блок пригоден для сбора среднебыстрых процессов в реальном времени, как напр. акустический сбор сигнала.

Schmidt, J.
Test Circuit AD31
Mikroprozessortechnik, Berlin 1 (1987) 3, pp. 76
The author explains the circuit principle, the most important internal functional groups, and the way of functioning of an 8-bits analog-to-digital converter being integrated monolithically in MOS technology and working according to the successive approximation method. The component is suited to the acquisition of real-time processes showing medium rapidity such as acoustical signals.

Löber, Chr.
RAM-Floppy – ein neues Speichermedium für Bürocomputer
Mikroprozessortechnik, Berlin 1 (1987) 3, S. 83
Virtuelle Disketten, d. h. Halbleiter-Zusatzspeicher, die sich aus der Sicht des Nutzers wie Disketten verhalten, gewinnen international an Bedeutung. Der Beitrag schildert die Anwendung und Eigenschaften von RAM-Floppys, die beiden möglichen Wirkprinzipien I/O-Mapping und Memory-Mapping und stellt schließlich eine nachnutzbare Lösung für den Bürocomputer A 5120 vor.

Löber, Chr.
ОЗУ на гибком Диске – новая запоминающая среда для конторских ЭВМ
Mikroprozessortechnik, Berlin 1 (1987) 3, стр. 83
Виртуальные дискеты, т. е. полупроводниковые дополнительные запоминающие устройства, которые с точки зрения потребителя ведут себя как дискеты, окажутся в центре международного внимания. В статье рассмотрены применение и свойства ОЗУ на гибких дисках, оба возможных принципа действия распределения устройства ввода-вывода и памяти и представлено используемое решение для конторской ЭВМ А 5120.

Löber, Chr.
RAM Floppy Disks – a New Storage Means for Office Computers
Mikroprozessortechnik, Berlin 1 (1987) 3, pp. 83
Virtual floppy disks, i. e. semiconductor supplementary stores behaving like floppy disks as seen by the user, become more and more important within the international scope. The author deals with the application and characteristics of RAM floppy disks, the both functioning principles possible, viz. I/O-mapping and memory mapping, and finally presents a solution ready for use adapted to the office computer A 5120.

Stuhec, H.; Vynal, D.; Rathmann, M.
KC 85/2 als intelligentes grafisches Display für den PC 1715
Mikroprozessortechnik, Berlin 1 (1987) 3, S. 86
Die Kopplung des PC 1715 mit dem Kleincomputer erlaubt, die Farbgrafikfunktionen des KC 85/2 direkt für den Personalcomputer zu nutzen. Der KC 85/2 wird in Verbindung mit einem Farbmonitor zum intelligenten Farbgrafikdisplay. Die Realisierung ist mit geringem Hardwareaufwand möglich, da nur zwei Koppelbaugruppen eingesetzt werden.

Stuhec, H.; Vynal, D.; Rathmann, M.
КС 85/2 как интеллектуальный графический дисплей для персональной ЭВМ 1715
Mikroprozessortechnik, Berlin 1 (1987) 3, стр. 86
Связь между персональной ЭВМ и мини-ЭВМ позволяет использование цветных графических функций КС 85/2 непосредственно для персональной ЭВМ. КС 85/2 в связи со цветным монитором станет интеллектуальным цветным графическим дисплеем. Реализация возможна с небольшим расходом на аппаратурное обеспечение ввиду того, что применяются только два элемента связи.

Stuhec, H,; Vynal, D.; Rathmann, M.:
KC 85/2 as Intelligent Graphical Display for the PC 1715
Mikroprozessortechnik, Berlin 1 (1987) 3, pp. 86
Coupling the PC 1715 with the minicomputer allows to use colour graphic functions of the KC 85/2 directly for the personal computer. In connection with a colour monitor the KC 85/2 becomes an intelligent colour graphical display. The realization is possible by means of low hardware expense since two coupling sub-assemblies are only used.

ZENIT '86

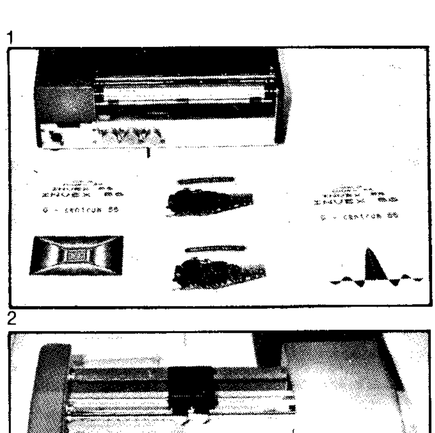
Wissenschaftlich-technisches Jugendschaffen in der ČSSR

Ende vergangenen Jahres fand im Kultur- und Informationszentrum der ČSSR in Berlin eine Ausstellung über den Beitrag der tschechoslowakischen Jugend zum wissenschaftlich-technischen Fortschritt statt. Die Veranstaltung stand unter dem Motto „Zenit 1986". Die „Zenit"-Bewegung unseres Nachbarlandes entspricht der Messe der Meister von morgen in der DDR.
In Berlin zu sehen waren 120 Exponate der verschiedensten Gebiete, ausgewählt aus den rund 2500 Arbeiten, die im Juni auf der zentralen „Zenit"-Messe in Prag vorgestellt wurden. U. a. wurden einige Druckermodelle gezeigt. Leider waren dazu nur wenige technische Daten zu erfahren.
Bild 1 zeigt den Plotter DIDAKTIK zur zweifarbigen grafischen Ausgabe von Informationen im A3- und A4-Format. Programmierbar ist er im XY-Koordinatensystem, dabei wird die y-Richtung durch den Papiervorschub realisiert. Die maximale Papiervorschubgeschwindigkeit beträgt 50 mm/s. Der Plotter kann auch über eine einfache Handsteuerung bedient werden.
Bild 2 zeigt einen Zweistiftdrucker, der über einen sehr einfachen Aufbau verfügt – z. B. wurde als Antrieb ein üblicher Motor, kein Schrittmotor, eingesetzt – und somit recht preiswert ist.
Wie bei allen Ausstellungen bzw. Veranstaltungen ähnlichen Charakters fanden auch hier die ausgestellten Computer das stärkste Interesse.
Auf runden Tischen angeordnet, standen sie jedem Besucher für das Erproben seiner Computerkenntnisse zur Verfügung. Im wesentlichen handelte es sich um Computer des Typs Ondra (siehe auch MP 1/87, Messebericht von der 28. Internationalen Maschinenbaumesse Brno) und PMD-85. Als Nachtrag zum Messebericht wollen wir noch ergänzen, daß gegenwärtig an einer LISP-Version für den Ondra gearbeitet wird. Eine PROLOG-Implementierung soll bereits vorliegen.
Eine Einstiegmöglichkeit in die Computertechnik auf etwas unkonventionelle Weise bot das Programmsystem KAREL. (Man könnte KAREL eigentlich auch als Programmiersprache bezeichnen.) Bei der Bezeichnung des Programms stand der Name des bekannten tschechischen Schriftstellers Karel Čapek Pate. Zugleich ist KAREL ein „Roboter", der sich auf dem Bildschirm des Computers bewegt und Befehle ausführen kann, die die Gestalt einfacher Worte bzw. Wortkombinationen haben. Dabei bewegt sich KAREL – ähnlich wie eine Figur auf dem Schachbrett – durch eine „Stadt". Die „Stadt" wird als Quadrat rechts auf dem Bildschirm dargestellt (Bild 3). Auf der linken Bildschirmseite werden die Befehle und Informationen für KAREL und die Hinweise, die der Computer zur Bedienung gibt, angezeigt.
Bei der Initialisierung des Programms verfügt KAREL über sogenannte Primitivbefehle. Das sind Hilfsbefehle (WORTSCHATZ, STADT, KORREKTUR, ZERLEGEN), Befehle für den „Roboter" (SCHRITT, LINKSUM, LEGE, NIMM) und Befehle, die das Bilden von zusammengesetzten und Bedingungsausdrücken ermöglichen (WIEDERHOLE, SOLANGE, WENN, ENDE). Die wenigen Anweisungen sind ausreichend, um KAREL zu programmieren. Beispielsweise kann er neue Wörter lernen, ein „Haus" bauen und eine „Mauer" errichten.
Das Programmsystem ermöglicht auch Schülern der unteren Klassen und natürlich Erwachsenen, die bisher über wenig Wissen auf diesem Gebiet verfügten, die Einführung in die Rechentechnik.

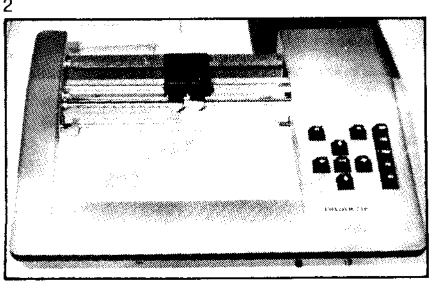
Ingo Paszkowsky

Fotos: Paszkowsky (3)

1



2



3



ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 3
S. 65–96

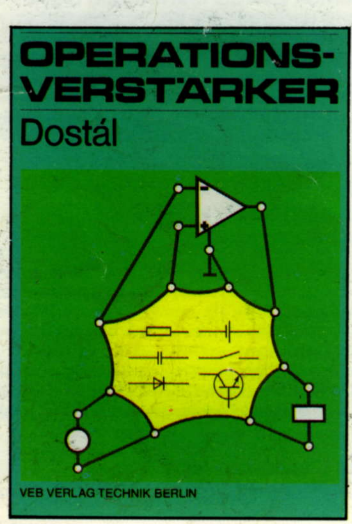
Neuerscheinungen und Nachauflage

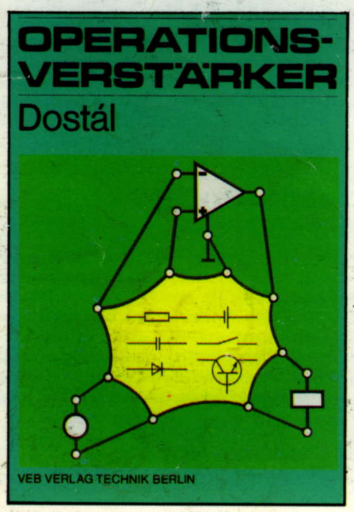


Bedienungs- und Instandhaltungsanleitungen
Inhalt – Form – Gestaltung

Von Prof. Dr. sc. techn. Georg-Wilhelm Werner und Dr.-Ing. Wolfgang Heyne. 1. Auflage. 128 Seiten, 160 Bilder, 20 Tafeln, Broschur, DDR 13,– M, Ausland 18,– DM. Bestellangaben: 553 442 8/Werner, Anleitungen

Die Autoren geben eine methodische Anleitung zur richtigen Inhaltswahl, rationellen Erarbeitung und zweckmäßigen Gestaltung aussagekräftiger Bedienungs- und Instandhaltungsanleitungen für technische Erzeugnisse. Checklisten und Sachwörtertabellen gewährleisten, daß beim Verfassen solcher Anleitungen vom „Was" und „Wie" nichts vergessen wird. Eine umfangreiche Falldiskussion zeigt gelungene und weniger gelungene Beispiele.





Taschenbuch Elektrotechnik in sechs Bänden
Band 2: Grundlagen der Informationstechnik

Herausgegeben von Prof. em. Dr. sc. techn. Dr. techn. h. c. Eugen Philippow. 3., stark bearbeitete Auflage. 984 Seiten, 836 Bilder, 161 Tafeln, Kunstleder, DDR 40,– M, Ausland 53,– DM. Bestellangaben: 553 615 9/Tb. Elektro 2

Die Auflage wurde völlig neu überarbeitet und auf den neuesten Stand gebracht. Darüber hinaus wurde sie um einige Abschnitte erweitert, die wichtige neue Berechnungsverfahren und in letzter Zeit besonders aktuell gewordene Gebiete zum Inhalt haben.

Operationsverstärker
Von Ing. Csc. Jiří Dostál. Aus dem Tschechischen. In deutscher Bearbeitung. 1. Auflage. 372 Seiten, 263 Bilder, 17 Tafeln, Leinen, DDR 38,– M, Ausland 56,– DM. Bestellangaben: 553 457 5/Dostál, Operation

Der Leser soll befähigt werden, nicht nur die bereits vorhandene Schaltungstechnik mit Operationsverstärkern zu beherrschen, sondern auch neue Entwicklungen zu analysieren. Der Autor stützt sich deshalb auf häufig anzutreffende Schaltungsgrundtypen und behandelt ausführlich deren so wichtige Spezialeigenschaften wie Spektral- und Zeitverhalten, Stabilität, Rauschen, Impedanzen u. a. sowie darüber hinaus die zur Erfassung dieser Parameter erforderliche Meßtechnik.

Das Buch kann als eines der umfassendsten Werke dieses Fachgebietes angesehen werden.



Auslieferung in diesen Tagen durch den Fachbuchhandel

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 4 · 1987



Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0233-2892

12. Mikroelektronik-Bauelemente-Symposium

Passende Chips

Grafik-Interface mit GDC U 82720

ISACAD – ein Entwurfssystem für bipolare digitale Gate-Array-Schaltkreise

Listing eines Steuerprogramms in PASCAL für den Grafik-Display-Controller U 82720

Lesen Sie dazu unseren Beitrag auf S. 99.

```
GDC-Beispielprogramme Vers 1.0 (c) Peter&Wolf  Seite 2


Const
{I/O-Adressen}  StatusPort=$1e;  DataPort=$1f; ZoomPort=$1b;
{Linienarten}   Continued=$ffff; Dashed=$cccc; DotDashed=$bbbb; Dotted=$aaaa;
{GDC-Kommandos} Wdat =$20; Curs =$49; Mask =$4a; Figs =$4c; Figd=$6c; PRam=$70;
                Zoom =$46;Synch =$0f; Pitch=$47; Unblk=$0d; Blk =$0c;
                CChar=$4b;Gchrd =$68; Vsync=$6e; Resetcode =$00;
{Schreibmodi}   PReplace=0; PXor=1; PReset=2; Pset=3;
{Figurentypen}  Line=8; Character=$10; Circle=$20; Rectangle=$40; Slanted=$80;
{Richtungen}    unten=0;rechtsunten=1;rechts=2;rechtsoben=3;
                oben =4;linksoben  =5;links =6;linksunten=7;
Type            SStrng                            : string[80];
Var             MaxX,MaxY,I,WordsPerLine          : Integer;
                ChSet                             : Array[0..127,0..7] of Byte;
                Zeichengenerator                  : File of Byte ;

{Kommandoausgabe an den GDC}
Procedure Command(C:Byte) ;
begin repeat until Port[StatusPort] and 2=0; Port[DataPort]:=C; end ;
{Parameterausgabe an den GDC}
Procedure Parameter(P:Byte) ;
begin repeat until Port[StatusPort] and 2=0; Port[StatusPort]:=P; end ;
{Setzen des Schreibzeigers}
Procedure SetXY(X,Y:Integer);
begin
     I:=(X shr 4) + (Y*WordsPerLine);
     Command(Curs); Parameter(lo(I)); Parameter(hi(I));
                    Parameter((X mod 16) shl 4);
end ;
{Setzen der Darstellbereiche Nr=0..1,X/Yanf=linke obere Ecke,Laenge=Zeilenzah}
Procedure Setber(Nr,Xanf,Yanf,Laenge:Integer);
begin
     I:=Xanf div 16+Yanf*WordsPerLine ;
     Command(PRam+(nr shl 2)); Parameter(lo(I)); Parameter(hi(I));
                               Parameter(16 * (Laenge mod 16));
                               Parameter((Laenge div 16) and $3f);
end ;
{Setzen der Zoomfaktoren fuer Darstellen und Schreiben 0..16}
Procedure Setzoom(Displayzoom,Writezoom:Byte);
begin
     Port[ZoomPort]:=Displayzoom; Command(Zoom);
     Parameter((Displayzoom shl 4 ) + Writezoom);
end ;
{Setzen der Schreibmaske Binaermuster fuer den naechsten Zugriff}
Procedure SetMask(Maske:Integer);
begin Command(Mask); Parameter(lo(Maske)); Parameter(hi(Maske)); end ;
{Setzen des Schreibmusters z.B. fuer gepunktete o. gestrichelte Linien}
Procedure Setpattern(Pattern:Integer);
begin Command(PRam+8); Parameter(lo(Pattern)); Parameter(hi(Pattern)); end ;
{Loechen des gesammten Schirms (mit 0) 64K Worte in 4 Schritten }
Procedure ClearScrean;
begin
     SetXY(0,0); SetMask(-1); {alle Bits werden gleichzeitig geschrieben}
     For I:=0 to 3 Do
         begin
            Command(Figs);Parameter(rechts); Parameter(-1); Parameter($3f);
            Command(Wdat+PReplace);Parameter(0); Parameter(0);
         end;
end ;
```

```
GDC-Beispielprogramme Vers 1.0 (c) Peter&Wolf  Seite 3


{Initialisiert den GDC entsprechen den angegeben Werten gueltig fuer die
 Beispielschaltung (16 MHz) und Fernsehnorm }
Procedure Init ;
Const AktiveWords= 40; HoriSyn=1; HoriFrontPorch= 12; HoriBackPorch=11;
      AktiveLines=280; VertSyn=1; VertFrontPorch= 2 ; VertBackPorch= 29;
    LinesPerCharRow=0; BlinkRate=3; CursorTopLine=0; CursorBottomLine=0;
      Mixed          = $00 ; Graphics       = $02 ; Character      = $20 ;
      Noninterlaced  = $00 ; Interlacedrpt  = $08 ; Interlaced     = $09 ;
      Norefresh      = $00 ; Refresh        = $04 ;
      Flash          = $00 ; Noflash        = $10 ;
      Master         = $00 ; Slave          = $01 ;
      Blank          = $00 ; UnBlank        = $01 ;
      NoCursor       = $00 ; Cursor         = $01 ;
      Blink          = $00 ; NoBlink        = $01 ; HighMem        =65536.0;
begin
  MaXY:=Trunc(HighMem/WordsPerLine)-WindowLines-1;
  MaxX:=16*WordsPerLine-1;
  Port[DataPort]:=Resetcode;
  Parameter(Graphics+NonInterlaced+Refresh+Noflash);
  Parameter(AktiveWords-2);
  Parameter(((VertSyn mod 8) shl 5) + HoriSyn -1);
  Parameter((VertSyn shr 3) +((HoriFrontPorch-1)shl 2));
  Parameter(HoriBackPorch-1); Parameter(VertFrontPorch);
  Parameter(lo(AktiveLines));
  Parameter(hi(AktiveLines)+(VertBackPorch shl 2));
  Setber(0,0,0,Lines);
  Command(Synch);
  Command(Vsync+Master);
  Command(CChar);
  Parameter(LinesPerCharRow+(Cursor shl 7));
  Parameter(((BlinkRate mod 4) shl 6) +(Blink shl 5)+CursorTopLine);
  Parameter((CursorBottomLine shl 3)+(BlinkRate div 4));
  Setzoom(0,0);
  Command(Pitch); Parameter(WordsPerLine);
  Command(Unblk);
end ;
{Zeichnet vom festgelegten Punkt (SetXY) einen Vektor der Laenge X,Y
 mit dem binaeren Muster Pattern im Schreibmode Wrmode}
Procedure Vektor(X,Y,Pattern,WRMode:Integer) ;
Var Tem,DD,Dir,DC,D,D1,D2:Integer;
begin
     Setpattern(Pattern); DC:=Abs(X); DD:=Abs(Y);
     if (DC<DD)  and (X>=0) and (Y>0)  then Dir := unten;
     if (DC>=DD) and (X>0)  and (Y>0)  then Dir := rechtsunten;
     if (Dc>DD)  and (X>0)  and (Y<=0) then Dir := rechts;
     if (DC<=DD) and (X>0)  and (Y<0)  then Dir := rechtsoben;
     if (DC<DD)  and (X<=0) and (Y<0)  then Dir := oben;
     if (DC>=DD) and (X<0)  and (Y<0)  then Dir := linksoben;
     if (DC>DD)  and (X<0)  and (Y>=0) then Dir := links;
     if (DC<=DD) and (X<0)  and (Y>0)  then Dir := linksunten;
     if DC<DD then begin Tem:=DC; DC:=DD; DD:=Tem; end;
     D :=(DD shl 1)-DC ;
     D2:=(DD-DC) shl 1 ;
     D1:=DD shl 1 ;
     Command(Figs); Parameter(Dir+Line);Parameter(lo(DC)); Parameter(hi(DC));
                    Parameter(lo(D));  Parameter(hi(D));
                    Parameter(lo(D2)); Parameter(hi(D2));
                    Parameter(lo(D1)); Parameter(hi(D1));
     Command(Wdat+WRMode );
     Command(Figd);
end ;
```

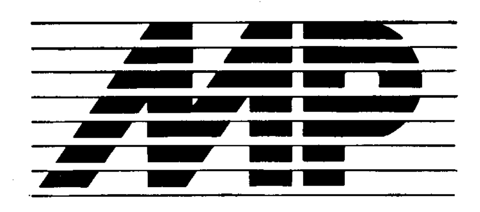
```
GDC-Beispielprogramme Vers 1.0 (c) Peter&Wolf  Seite 4


{Zeichnet einen Punkt bei X,Y im angegeben Writemode (Set,Reset,Xor,Replace)}
Procedure Plot(X,Y,WRMode:Integer);
begin
     SetXY(X,Y); Command(Wdat+WRMode); Command(Figs); Parameter(rechts);
     Command(Figd);
end;
{Zeichnet einen Grafikkursor auf die Position X,Y}
Procedure Cursor(X,Y:Integer);
begin
     Command(Wdat+PXor);
     SetXY(X+3,Y-3);
     Command(PRam+8); Parameter($0); Parameter($8); Parameter($8);
                      Parameter($8); Parameter($7f);Parameter($8);
                      Parameter($8); Parameter($8);
     Command(Figs);   Parameter(Character+oben);Parameter(7);
     Command(Gchrd);
end ;
{Zeichnet eine Linie von X1,Y1 nach X2,Y2 mit Pattern und Writemode}
Procedure GdcDraw(X1,Y1,X2,Y2,Pattern,WRMode:Integer);
begin SetXY(X1,Y1); Vektor(X2-X1,Y2-Y1,Pattern,WRMode); end;
{Zeichnet einen Buchstaben auf die aktuelle Position mit Groesse von
 1*6 ..16*6 und mit den Richtungen 0..7 und 128..135 (geneigt)}
Procedure Gdcw(C:char;WRMode,Groesse,Richtung:Integer);
Var I:Integer;
   begin
     Setzoom(0,(Groesse div 6)-1);
     Command(Wdat+WRMode);
     Command(PRam+8); For I:=0 to 7 Do Parameter(ChSet[ord(C),I]);
     Command(Figs); Parameter(Character+Richtung) ; Parameter(7);
     Command(Gchrd);
   end;
{Schreibt einen String auf die Position XSt,YSt mit Groesse und Richtung
 wie oben}
Procedure Gdctext(XSt,YSt,Groesse,Richtung.Integer; S:Sstrng; WRMode:Integer);
Var X,Y:Integer;
begin
     X:=XSt; Y:=YSt;
     For I:=1 to Length(S) Do
         begin
              SetXY(X,Y);
              Gdcw(S[I],WRMode,Groesse,Richtung);
              case Richtung and 7 of
                0: Y:=Y-Groesse;
                1: begin X:=X+Groesse; Y:=Y-Groesse; end;
                2: X:=X+Groesse;
                3: begin X:=X+Groesse; Y:=Y+Groesse; end;
                4: Y:=Y+Groesse;
                5: begin X:=X-Groesse; Y:=Y+Groesse; end;
                6: X:=X-Groesse;
                7: begin X:=X-Groesse; Y:=Y-Groesse; end;
              end;
         end;
end;
```

```
GDC-Beispielprogramme Vers 1.0 (c) Peter&Wolf  Seite 5


{Zeichnet einen Kreis mit dem Mittelpunkt X,Y }
Procedure GdcCircle(X,Y,Radius,Pattern,WRMode:Integer);
Const Eins_Durch_Wurzel_Zwei=0.707106 ;
Var DC:Integer;
 Procedure Setwer;
 begin
     Parameter(lo(DC));                 Parameter(hi(DC));
     Parameter(lo(Radius-1));           Parameter(hi(Radius-1));
     Parameter(lo((Radius-1) shl 1));   Parameter(hi((Radius-1) shl 1));
     Parameter(-1);                     Parameter(-1);
     Parameter(0);                      Parameter(0);
     Command(Figd);
 end;
begin
     Setpattern(Pattern);
     dc:=round(Eins_Durch_Wurzel_Zwei*Radius);
     Command(Wdat+WRMode );
     SetXY(X-Radius,Y);Command(Figs);Parameter(Circle+unten      );Setwer;
     SetXY(X,Y+Radius);Command(Figs);Parameter(Circle+rechtsunten);Setwer;
     SetXY(X,Y-Radius);Command(Figs);Parameter(Circle+rechts     );Setwer;
     SetXY(X-Radius,Y);Command(Figs);Parameter(Circle+rechtsoben );Setwer;
     SetXY(X+Radius,Y);Command(Figs);Parameter(Circle+oben       );Setwer;
     SetXY(X,Y-Radius);Command(Figs);Parameter(Circle+linksoben  );Setwer;
     SetXY(X,Y+Radius);Command(Figs);Parameter(Circle+links      );Setwer;
     SetXY(X+Radius,Y);Command(Figs);Parameter(Circle;linksunten );Setwer;
end;
{Zeichnet ein Rechteck mit den Diagonalenpunkten X0,X1,Y0,Y1}
Procedure GdcReck(X0,Y0,X1,Y1,Pattern,WRMode:Integer);
Var A,B,tem:Integer;
begin
     if X0>X1 then begin tem:=X1; X1:=X0 ;X0:=tem; end;
     if Y0>Y1 then begin tem:=Y1; Y1:=Y0 ;Y0:=tem; end;
     Setpattern(Pattern); SetXY(X0,Y0);
     B:=Y1-Y0;      A:=X1-X0;
     Command(Wdat+WRMode);
     Command(Figs); Parameter(rectangle+2);Parameter(3);Parameter(0);
                    Parameter(lo(A)); Parameter(hi(A));
                    Parameter(lo(B)); Parameter(hi(B));
                    Parameter(-1);    Parameter(-1);
                    Parameter(lo(A)); Parameter(hi(A));
     Command(Figd);
end ;
{Initialisiert den GDC,Loescht den Bildschirm und Zeichnet die Umrandung}
Procedure InitGraf;
begin
    WordsPerline:=64; Init; ClearScrean;
    Assign(ZeichenGenerator,'CHSET.DUM'); Reset(ZeichenGenerator);
    For X:=0 to 127 Do
       For Y:=0 to 7 Do Read(ZeichenGenerator,Chset[X,Y]);
    GdcDraw(0,0,MaxX,0,Continued,Pset);
    GdcDraw(0,0,0,MaxY,Continued,Pset);
    GdcDraw(0,MaxY,MaxX,MaxY,Continued,Pset);
    GdcDraw(MaxX,0,MaxX,MaxY,Continued,Pset);
end;
```

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0233–2892



Mikroprozessortechnik, Heft 4 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 2 87 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)

Titel: Keller

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

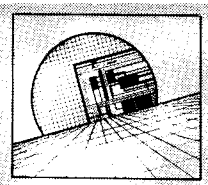
Redaktionsschluß: 17. Februar 1987

AN (EDV) 49837

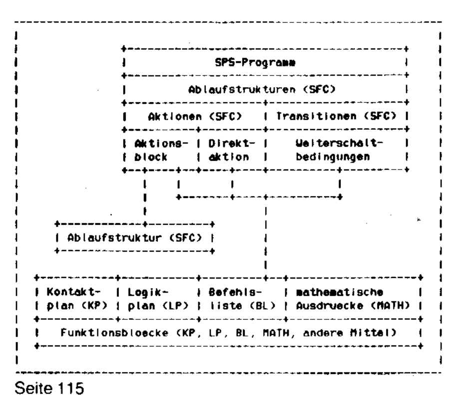
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustředni Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavacko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hanoi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010, und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



MEBS 1987
Zum 12. Mal findet in Frankfurt (Oder) das Mikroelektronik-Bauelementesymposium (vormals Halbleitersymposium) statt – seit 1985 unter der Schirmherrschaft des Ministers für Elektrotechnik und Elektronik.
Immer wichtiger wird die rechtzeitige, gezielte und umfassende Information der Erzeugnis- und Verfahrensentwickler, der Konstrukteure, Technologen und Applikationsingenieure über Eigenschaften, Kenndaten und Einsatzmöglichkeiten der verfügbaren Bauelemente. Dieser gestiegenen Bedeutung und der großen Nachfrage entsprechend schließt sich dem 12. MEBS (11. bis 13. Mai 1987) eine Fachtagung mit ausgewählten Vorträgen des Symposiums (14. bis 15. Mai 1987) an. Aus Anlaß des Symposiums werden im ersten Teil dieses Heftes Beiträge zu der Thematik veröffentlicht. Zwei der Entwicklungen – das ISACAD-System und das 16-Bit-Mikroprozessorsystem U8000 – dürften dem Leser nicht mehr ganz unbekannt sein. Neu hingegen ist der Grafik-Display-Controller U 82720 D, der in Mikrorechnersystemen eine leistungsfähige grafische Datenverarbeitung ermöglicht.
Informationen zu weiteren auf dem 12. MEBS in Vorträgen vorgestellten Bauelementen der Computertechnik finden Sie in den folgenden Ausgaben der Mikroprozessortechnik.



Seite 115

a_i | x^2 | *bisheriger Stack*

Σ_i — *Gleitpunkt-Akkumulator*

abzuarbeitende Formel: $y = \sum_{i=0}^{n} a_{2i+1} x^{2i+1}$

$= (\ldots((a_{2n+1} x^2 + a_{2n-1}) x^2 + a_{2n-3}) x^2 + \ldots + a_1) x$

Seite 120

Inhalt

Vorschau

Im Heft 5/1987 können Sie u. a. lesen:
– Fertigungsorientierte Meß- und Prüftechnik – Stand und Tendenzen
– Mikroprozessorkompatibler D/A-Wandler C 560 D
– C 574 C – schneller, kompletter 12-Bit-A/D-Wandler mit Mikroprozessorinterface

Lizenzen aus der Ungarischen VR

Über Neuerungen, Erfindungen, Patente und Lizenzangebote aus der UVR informierte eine Veranstaltung mit dem Titel TECHNO – INFORM. Fachvorträge ergänzten die dreitägige Ausstellung im Januar im Haus der Ungarischen Kultur in Berlin, die vom Industriellen Unternehmen für Reklame und Werbung und vom Lizenzia Außenhandelsunternehmen zur Verwertung von Erfindungen und Innovationen organisiert wurde.
Auf zwei der vorgestellten Lösungen soll etwas näher eingegangen werden.
Bild 1 zeigt den Video-Prozessor DCD VP-01 mit Konfiguration. Der Prozessor kann mit Display und Tastatur eigenständig arbeiten oder als intelligentes Peripheriegerät an einen übergeordneten Rechner angeschlossen werden. VP-01 beinhaltet A/D-Wandler, DMA-Steuereinheit, Mikrorechner, Speicher, D/A-Wandler und Schnittstellen. Der D/A-Wandler digitalisiert die auf seinen Eingang ankommenden Videosignale. Er arbeitet mit einer Bildauflösung von 300×400 Punkten. Maximal 64 Grauwertstufen können eingestellt werden. Der Mikrorechner basiert auf einem Z80-Mikroprozessor. Zum System mitgelieferte Software ermöglicht die Bearbeitung des gespeicherten Bildes, wie Ausgeben eines Ausschnitts oder Filtern bestimmter Grauwertstufen. Bisher ist nur die Verarbeitung von Schwarzweißbildern möglich, vorgesehen ist aber, in Kürze ein System für die farbige Bildverarbeitung anzubieten. Und zwar soll ein entsprechender Videoprozessor, der VP-02, in die Produktion überführt werden.
Wege zur Datenübertragung, z. B. über Hochspannungsleitungen oder über das 220-V-Netz, zeigten die Kohlenbergwerke Oroszlány auf. Für diese Art der Informationsübertragung wird das Trägerfrequenzprinzip verwendet. Über eine Reihe von Lösungen für verschiedene Übertragungsaufgaben informierte das Unternehmen in einem Vortrag, so u. a. über die Datenübertragung mit Trägerfrequenz-MODEMs über das Fernsprechnetz unter Beibehaltung des Sprechbetriebes. Auf der Ausstellung wurde die Verbindung zweier Computer über das 220-V-Netz demonstriert. Einen Computer mit den zur Kopplung notwendigen Einrichtungen ist im Bild 2 zu sehen. Die Informationsübertragung geschieht im Duplex-Betrieb. Dabei beträgt die Übertragungsrate maximal 1200 Baud. Die verwendeten Frequenzen sind 42 bzw. 63 kHz. Ein Entwicklungsziel der Kohlenbergwerke Oroszlány ist die Realisierung lokaler Netze auf Starkstromnetzen. *MP*

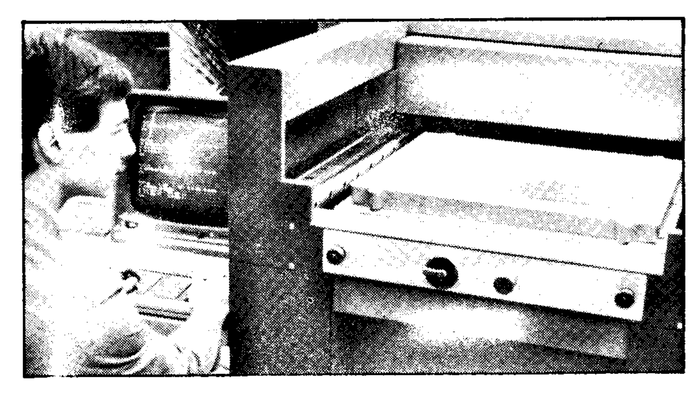
Neue Computerzeitschrift in Ungarn

Unter dem Titel „Computerworld – Szamitastechnika" (Rechentechnik) gibt es in Ungarn eine neue Zeitschrift. Herausgeber ist die ungarisch-amerikanische Joint venture Computerworld Informatika GmbH, die 1986 von zwei ungarischen Buchverlagen sowie dem rechentechnischen Verlag Computerworld Communications aus den USA gegründet wurde. Das Gemeinschaftsunternehmen befaßt sich in Ungarn mit der Herausgabe von Periodika und wissenschaftlichen Fachbüchern. Die Zeitschrift gab es 1986 monatlich, ab 1987 erscheint sie alle zwei Wochen.

Aus Budapester Rundschau

Leiterplattenprüfsystem

Mit dem Leiterplattenprüfsystem Sitest 550 von Siemens lassen sich unbestückte Leiterplatten, Innenlagen und Multilayer elektrisch prüfen. Das System läßt sich an Prüflinge unterschiedlichen Formats und ständig wechselnder Prüfpunkte-Anordnung anpassen. Der Sitest 550 prüft sämtliche Verbindungen auf der Leiterplatte auf Durchgangs- bzw. Isolationswiderstand, macht automatisch eine Gut-Schlecht-Sortierung und hält dabei Schritt mit den Taktzeiten eines automatisierten Fertigungslaufs.


Werkfoto

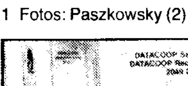
Kristalle für optisch gesteuerte Schaltungen

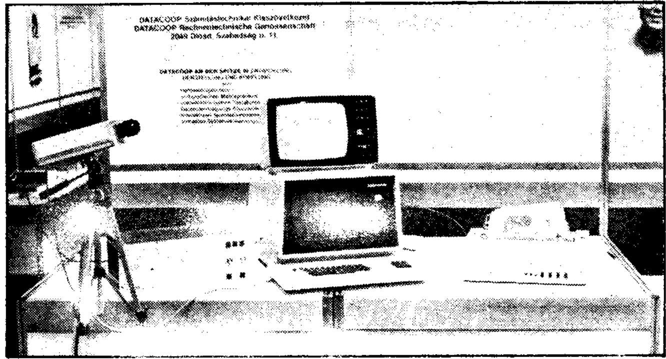
Kaliumniobatkristalle werden im Rahmen eines Forschungsprojekts gegenwärtig mit Erfolg im Labor für Festkörperphysik der Technischen Hochschule Zürich gezüchtet. Die Herstellung dieser hochwertigen künstlichen Kristalle stelle nach Ansicht der Forscher „einen Markstein in der Entwicklung volloptischer Systeme dar", berichtet die „Basler Zeitung".
Dadurch könnten möglicherweise die elektronischen durch optisch gesteuerte Schaltungen, Computer und Fernmeldesysteme abgelöst werden. Im Platintiegel und in 1 065 Grad heißer Schmelze wachsen unter stetem Zug innerhalb von drei Wochen bis zu 30 Gramm schwere Kristalle heran, schreibt das Blatt. Im Vergleich zu natürlich gewachsenen Kristallen zeichneten sich im Labor gezüchtete Silizium- und Quarzkristalle durch eine einheitliche Struktur aus. In Zürich sei es gelungen, Kaliumniobatkristalle in einer weltweit bisher unerreichten Reinheit herzustellen.
Sie verfügen über sehr gute Eigenschaften für die Optoelektronik, das heißt für die Informationsverarbeitung mit Licht und elektrischem Strom. *ADN*

Termine

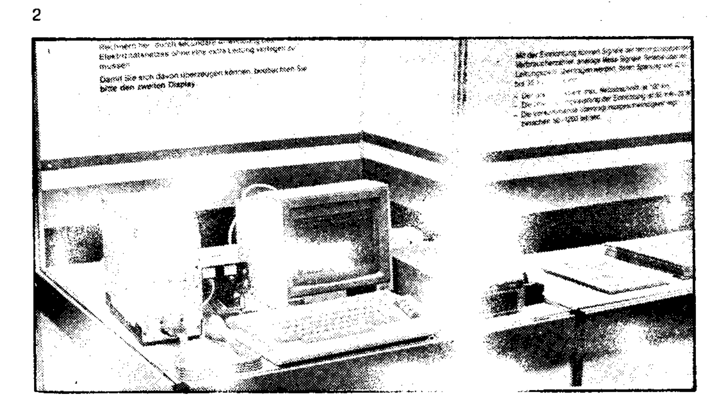
Computer- und Mikroprozessortechnik '87
WER? Wissenschaftliche Sektion Computer- und Mikroprozessortechnik in Zusammenarbeit mit dem Bezirksvorstand Magdeburg der KDT und der TU Magdeburg, Sektion Automatisierungstechnik/Elektrotechnik
WANN? 8. und 9. Dezember 1987
WO? Magdeburg
WAS?
- Prozeßperipherie
- Anwendung der Gate-Array-Technik
- 16-Bit-Rechner
- Bussysteme

WIE? Vortrags- und Teilnehmeranmeldungen an:
Technische Universität „Otto von Guericke" Magdeburg, Sektion Automatisierungstechnik/Elektrotechnik, Prof. Dr. sc. techn. Seifart, PSF 124, Magdeburg, 3010

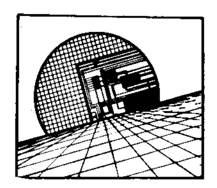

1 Fotos: Paszkowsky (2)



2



Grafik-Interface mit dem U 82720



Mario Bankel, Peter Brückner,
René Wolf,
VEB Mikroelektronik „Karl Marx"
Erfurt

Der Einsatz des Grafik-Display-Controllers GDC U 82720 ermöglicht eine leistungsfähige grafische Datenverarbeitung in Mikrorechnersystemen. Nachfolgend wird eine Einführung zum GDC gegeben. Eine Applikationsschaltung und das dazugehörige PASCAL-Steuerprogramm ermöglichen in Systemen mit SCP-Betriebssystem den sofortigen Einstieg in die Grafik-Welt. Es wird eine Variante mit einem 64 KWorte (ein Megapixel) großen Bildwiederholspeicher vorgestellt. Die Bildformatgröße ist frei programmierbar. Im PASCAL-Beispielprogramm werden 640 × 280 Bildpunkte auf einer Zeichenfläche von 1024 × 1024 Pixelelementen abgebildet. Spezielle Funktionen des GDC wie Zooming und Panning können mit der vorgestellten Hardwarekonfiguration benutzt werden. Als Display eignen sich Fernseh-Monitore mit einer Videobandbreite > 10MHz.

Signal	Pin	Pin	Signal
2 x WCLK	1	40	U_{CC}
DBIN	2	39	A 17
HSYNC	3	38	A 16
V/EXT SYNC	4	37	AD_{15}
BLANK	5	36	AD_{14}
ALE	6	35	AD_{13}
DRQ	7	34	AD_{12}
DACK	8	33	AD_{11}
RD	9	32	AD_{10}
WR	10	31	AD_9
A0	11	30	AD_8
DB_0	12	29	AD_7
DB_1	13	28	AD_6
DB_2	14	27	AD_5
DB_3	15	26	AD_4
DB_4	16	25	AD_3
DB_5	17	24	AD_2
DB_6	18	23	AD_1
DB_7	19	22	AD_0
GND	20	21	LPEN

Bild 1 Anschlußbelegung und Schaltzeichen des GDC U 82720

Einführung

Die nachfolgenden einführenden Bemerkungen stellen ein stark komprimiertes Minimalwissen zum GDC U 82720 dar. Ausführlich wird der GDC in /1/ beschrieben.

Die Darstellung und Bearbeitung von Grafiken auf einem hochauflösenden Bildschirm sollte mit der höchstmöglichen Verarbeitungsgeschwindigkeit erfolgen. Dies wird durch Grafikcontroller in Verbindung mit (möglichst) mehreren Speicherebenen für den Bildwiederholspeicher und externer Zusatzhardware zur Datenserialisierung am Videointerface erreicht.

Für eine hochauflösende Grafikdarstellung im Rasterverfahren wird dabei sehr viel Speicherplatz benötigt. Bei farbiger Darstellung sind in jedem Fall mehrere Speicherebenen erforderlich. Hierbei erhöht sich der Speicherbedarf um ein Vielfaches.

Ein separater, vom Hauptspeicher unabhängiger Videospeicher (Bildwiederholspeicher) wird direkt vom Grafikcontroller verwaltet. Der steuernde Mikroprozessor (oder Rechner) gibt dann nur noch Instruktionen an den zwischengeschalteten Grafikprozessor, der diesen Videospeicher verwaltet.

Der Aufwand für die Erarbeitung der Steuersoftware für ein grafisches Display kann durch den komfortablen Befehlssatz des GDC und die verfügbaren Zeichnungsalgorithmen zum Zeichnen grafischer Darstellungen (z. B. Kreise oder Rechtecke) klein gehalten werden. Für die Modifikation eines Bildpunktes benötigt der GDC einen Read-Modify-Write-Zyklus, der vier Taktperioden dauert. Bei einer Darstellung (Figur) mit 100 Pixeln ist dieser Prozeß bei einer internen GDC-Taktfrequenz von 2 MHz nach 200 Mikrosekunden abgeschlossen.

Charakteristik

- Mikroprozessor-Interface; DMA-Transfer in Verbindung mit DMA-Controllern möglich
- Kommandospeicher in GDC-internem FIFO-Buffer
- Bildwiederholspeicher, adressierbar 256 KWorte zu je 16 Bit; Zugriffsmöglichkeit über Read-Modify-Write-Zyklus (RMW) und Display-Zyklus ohne RMW
- Eingang für Lichtstift
- Möglichkeit der externen Video-Synchronisation
- Grafik-Betriebsart (Graphics Mode) programmierbar; Verwaltung von 4 MBit Bildwiederholspeicher
- Möglichkeiten bei der Grafikdarstellung:
Zeichnen von Geraden, Bögen und Kreisen, Rechtecken und grafischen Zeichen mit einer Punktfolgefrequenz von 32 MHz in der normalen bzw. 64 MHz in der gedehnten Darstellungsart (sog. Wide-Display-Mode); zwei unabhängig voneinander rollbare Bildfenster
- Zeichen-Betriebsart (Character Mode) programmierbar; Verwaltung eines Bildwiederholspeichers mit 8 K × 13 Bit Character- und Attributspeicher
- Möglichkeiten bei der Zeichendarstellung:
automatische Verschiebung des Cursors; vier unabhängig voneinander rollbare Flächen; programmierbare Cursorgröße; 256 Zeichen pro Zeile; bis zu 100 Zeilen je Bildebene
- Gemischte Grafik- und Zeichen-Be-

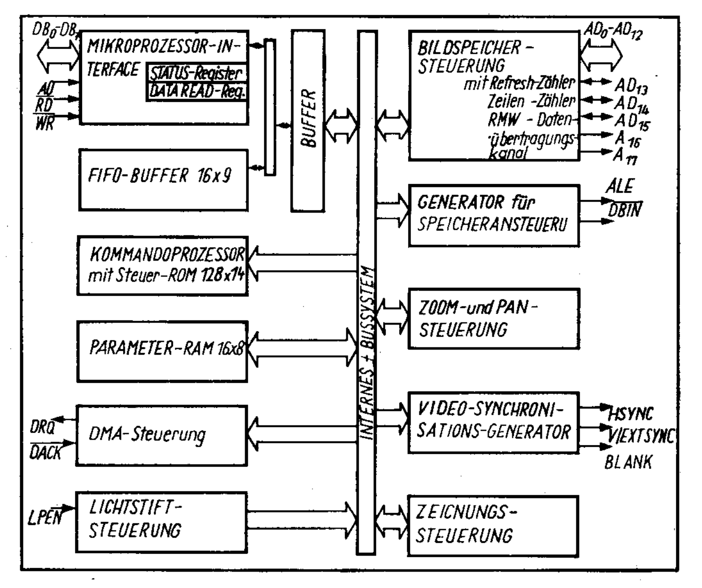


Bild 2 Blockschaltbild des GDC U 82720

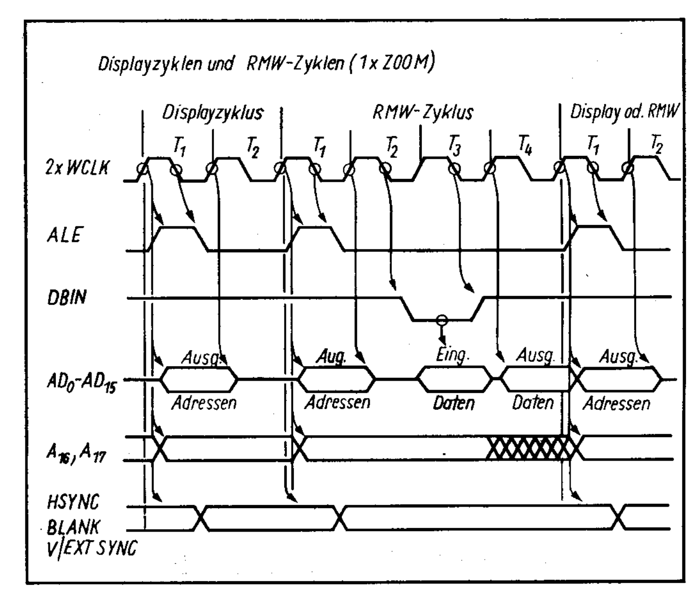


Bild 3 Zeitverhalten des GDC bei 1 × ZOOM

triebsart (Mixed Graphics and Character Mode) programmierbar; Verwaltung von 64 K Zeichen-Bildwiederholspeicher und ein Megapixel Grafik-Bildwiederholspeicher

- Vergrößerungsfaktor (Zoom) 1... 16fach; Schwenkfunktion (Panning); programmierbare Parameter für das Videoraster
- DMA-Eigenschaften: Übertragung von Byte oder Wort in vier Taktperioden je Byte-Übertragung
- Betriebsspannung +5 V
- 40-Pin-DIL-Gehäuse (2,54-mm-Raster)
- Kompatilität INTEL 82720, NEC 7220

Tafel 1 Funktionskurzbeschreibung der Anschlüsse des GDC U 82720

Symbol	Art	Benennung und Funktionskurzbeschreibung	
2xWCLK	Eingang	Takteingang	
$\overline{DBIN}$	Ausgang	Buseingangssignal als Strobesignal zum Einlesen der Daten des Bildwiederholspeichers in den GDC	
HSYNC	Ausgang	Horizontalsynchronisationssignal zur Erzeugung des Zeilenrücklaufs auf dem Display	
V/EXT SYNC	Ein-/ Ausgang	Vertikalsynchronisationssignal zur Erzeugung des Bildrücklaufs auf dem Display (Ausgangssignal); in der GDC-Slave-Betriebsart als Eingangssignal zur Synchronisation des Bildrasterzeitverhaltens mit einem Master-GDC	
BLANK	Ausgang	Video-Austastsignal	
RAS (ALE)	Ausgang	Speicherzyklussignal für Zeilenadreßübernahme (RAS-Signal) für dynamische RAMs oder Demultiplexsignal (ALE-Signal – Address Latch Enable) für den Adreß-/Datenbus mit statischen RAMs im Bildwiederholspeicher	
DRQ	Ausgang	DMA-Anforderungssignal (DMA Request) zur Steuerung eines externen DMA-Controllers	
$\overline{DACK}$	Eingang	DMA-Bestätigungssignal (DMA Acknowledge) zur Steuerung eines externen DMA-Controllers	
$\overline{RD}$	Eingang	Strobesignal (Read) zum Einlesen von GDC-Daten in den System-Mikroprozessor	
$\overline{WR}$	Eingang	Strobesignal (Write) zum Einlesen von Mikroprozessor-Daten in den GDC	
A0	Eingang	Auswahl der Adresse zur Unterscheidung von Kommandos und Lese-/ Schreibdaten	
$DB_0 \ldots DB_7$	Ein-/ Ausgänge	bidirektionaler Mikroprozessor-Datenbus des GDC; Bussteuerung erfolgt mit den Signalen $\overline{WR}$ für Eingabe und $\overline{RD}$ für Ausgabe	
GND	–	Masseanschluß	
LPEN	Eingang	Lichtstiftsignal (Light Pen Detect)	
$AD_0 \ldots AD_{12}$	Ein-/ Ausgänge	Adreß-/Datenbus des Bildwiederholspeichers (Bit 0 ... 12)	
$AD_0 \ldots AD_{15}$	Ein-/ Ausgänge	Funktion wird entsprechend der Betriebsart des GDC festgelegt:	
		– bei Grafik-Betriebsart	→ Adreß-Datenbus des Bildwiederholspeichers (Bit 13 ... 15)
		– bei Zeichen-Betriebsart	→ Zeilenzählerausgänge (Bit 13 ... 15)
		– bei Misch-Betriebsart	→ Adreß-/Datenbus des Bildwiederholspeichers (Bit 13 ... 15)
A16	Ausgang	Funktion wird entsprechend der Betriebsart des GDC festgelegt:	
		– bei Grafik-Betriebsart	→ Adreßbus des Bildwiederholspeichers (Bit 16)
		– bei Zeichen-Betriebsart	→ Zeilenzählerausgang (Bit 3)
		– bei Misch-Betriebsart	→ Attributblinksignal und Rücksetzen des Zeilenzählers
A17	Ausgang	Funktion wird entsprechend der Betriebsart des GDC festgelegt:	
		– bei Grafik-Betriebsart	→ Adreßbus des Bildwiederholspeichers (Bit 17)
		– bei Zeichen-Betriebsart	→ Cursor- und Zeilenzählerausgang (Bit 4)
		– bei Misch-Betriebsart	→ Cursorausgang und Flaganzeige für Bildbetriebsart
U_{CC}	–	Betriebsspannungsanschluß +5 V	

Bild 1 zeigt die Anschlußbelegung des GDC U82720. Auf Tafel 1 kann eine Funktionskurzbeschreibung der Anschlüsse entnommen werden. Zur Verdeutlichung des inneren Aufbaus ist im Bild 2 das Blockschaltbild des U82720 dargestellt. Nachfolgend eine kurze Beschreibung der Elemente des Blockschaltbildes:

Mikroprozessor-Interface

Das Mikroprozessor-Interface mit den Datenbus-Ein/Ausgängen $DB_0 \ldots DB_7$ ist acht Bit breit und bidirektional ausgeführt. Zum Interface gehören das Statusregister und das Data-Read-Register. Ein zusätzlicher FIFO-Puffer hat die Organisation 9 × 16 Bit.

Kommandoprozessor

Die Inhalte des FIFO-Puffers werden mit dem Kommandoprozessor (Command Processor) interpretiert. Dabei erfolgen eine Dekodierung der im FIFO befindlichen Kommandobytes und die Verteilung der Befehlsparameter an ihre Bestimmungsorte im GDC.

DMA-Steuerung

Die DMA-Steuerung (DMA-Control) des GDC übernimmt die Datenübertragung am Mikroprozessor-Businterface, wenn mit einem DMA-Controller kommuniziert wird. Die Steuersignale für DMA-Anforderung, DREQ (DMA-Request), sowie DMA-Bestätigung, $\overline{DACK}$ (DMA-Acknowledge), können im Handshake-Betrieb für die Ansteuerung beliebiger DMA-Controller (z. B. UA858D, i8257, i8237) verwendet werden.

Parameter-RAM

Dieser 16 Byte tiefe RAM speichert Befehlsparameter, die während des Darstellungs- und Zeichenprozesses wiederholt abgefragt werden. In der Zeichen-Betriebsart enthält der RAM die Display-Flächenaufteilungsparameter. In der Grafik-Betriebsart werden die Zeichenpattern und Grafikelemente gespeichert.

Video-Synchronisationsgenerator

Der Generator der Video-Synchronisationslogik des GDC erzeugt, basierend auf der Taktfrequenz, das Rasterzeitverhalten für fast alle Zeilensprung- und Nichtzeilensprungformate sowie für das Wiederholungsfeld beim Zeilensprungformat.

Generator für Speicheransteuerung

Ein Generator für Erzeugung des Speicherzeitverhaltens ermöglicht die Bereitstellung von zwei Speicherzyklustypen; einen zoombaren 2-Takt-Display-Zyklus sowie einen Read-Modify-Write-Zyklus (RMW) mit vier Taktzyklen. Als Signale zur Ansteuerung des Bildwiederholspeichers stehen die Speicherzyklussignale RAS (ALE) und DBIN zur Verfügung.

Zoom- und Pan-Steuerung

Der GDC ermöglicht unter Einsatz minimaler externer Hardware die Anwendung der programmierbaren Vergrößerung der Darstellung auf dem Bildschirm. Gleichfalls ist eine Schwenkbewegung über die für die Bilddarstellung benutzte Display-Bildfläche möglich. Bild 3 zeigt das Zeitverhalten für 1 × ZOOM. Eine einfache Variante zur hardwaretechnischen Zoom-Realisierung zeigt Bild 6.

Zeichnungs-Steuerung

Der Zeichnungsprozessor dieser Steuerung enthält die für die Berechnung der Pixeladressen und -positionen von diversen grafischen Darstellungen notwendige Logik. Außer einem gegebenen Startpunkt und entsprechenden Zeichnungsparametern benötigt der Zeichnungsprozessor keine weitere Unterstützung, um die Darstellung vollständig zu zeichnen.

Bildwiederholspeicher-Steuerung

Mit dieser Steuerung erfolgt das Multiplexen der Adreß- und Dateninformationen von und zum Bildwiederholspeicher des GDC. Die Steuerung enthält 16-Bit-Logikeinheiten, um den Inhalt des Bildwiederholspeichers während der RMW-Zyklen ändern zu können. Weiterhin erfolgt die Ansteuerung des Zeilenzählers in der Zeichen-Betriebsart und des Refresh-Zählers für dynamische RAMs im Bildwiederholspeicher.

Lichtstift-Steuerung

Für den Fall, daß am Lichtstifteingang während zweier steigender 2×WCLK-Flanken H-Pegel anliegt, wird das Lichtstiftregister mit der aktuellen gültigen Adresse geladen. Ein Statusbit des GDC zeigt danach dem System-Mikroprozessor an, daß das Lichtstiftregister eine gültige Adresse enthält. In der nachfolgenden Applikationsschaltung wird auf die Verarbeitung einer Lichtstiftinformation aus Aufwandgründen verzichtet.

Funktionelle Besonderheiten des U 82720

Zur hard- und softwaretechnischen Einbindung des GDC werden lediglich zwei Adressen zur Adressierung des Statusregisters und des FIFO-Puffers benötigt. Die Unterscheidung erfolgt mit dem Adreßbit A0. Im Bild 4 ist der Aufbau der Mikroprozessor-Businterface-Register dargestellt. Der U 82720 besitzt 20 Kommandos. Jedes Kommando ist ein in 8 Bit kodiertes Kommandobyte. Eine ausführliche Erläuterung der einzelnen Kommandos ist in /1/ enthalten. Die GDC-Kommandos können gemäß Tafel 2 unterteilt werden. Obwohl die GDC-interne Steuerung des Bildwiederholspeichers während der Rasterabtastung die Adressen für den Bildwiederholspeicher erzeugt, ist sie nicht unmittelbar an den Pixeloperationen zum Bildaufbau

Bild 4 Aufbau des GDC-internen Mikroprozessor-Businterface-Registers

A0	Lese-Operation	Schreib-Operation
0	Status-Register	Parameter in FIFO
1	FIFO-lesen	Kommando in FIFO

Tafel 2 Befehlsübersicht des GDC

VIDEO-STEUERKOMMANDOS (Video Control Commands)		
RESET	(Reset Command) Rücksetzen des GDC in den Freilaufbetrieb (Idle State)	
SYNC	(Sync Format Specify Command) Festlegung des Formats der Displayfläche	
CCHAR	(Cursor and Character Characteristics Command) Festlegung der Spaltenhöhe für Cursor und Zeichen	
DARSTELLUNGS-STEUERKOMMANDOS (Display Control Commands)		
START	(Start Command) Beendigung des Freilaufbetriebs und Freigabe des Displayschirms	
BCTRL	(Display Blanking Control Command) Steuerung der Freigabe zur Dunkeltastung des Displays	
ZOOM	(Zoom Factor Specify Command) Festlegung des Vergrößerungsfaktors für die grafische Darstellung	
CURS	(Cursor Position Specify Command) Steuerung der Position des Cursors im Bildwiederholspeicher	
PRAM	(Parameter RAM Load Command) Definition der Startadresse, der Länge(n) der Displayfläche(n), sowie der acht Bytes für ein grafisches Zeichen	
PITCH	(Pitch Specification Command) Festlegung der Breite des Bildwiederholspeichers in X-Richtung	
ZEICHNUNGS-STEUERKOMMANDOS (Drawing Control Commands)		
WDAT	(Write Data Command) Einschreiben von Datenwörtern bzw. -bytes in den Bildwiederholspeicher	
MASK	(Mask Register Load Command) Festlegung des Inhalts des Maskenregisters	
FIGS	(Figure Drawing Parameters Specify Command) Festlegung der Parameter für den Zeichnungsprozessor	
FIGD	(Figure Draw Start Command) Zeichnung der vorher festgelegten Darstellung	
GCHRD	(Graphics Character Draw and Area Filling Start Command) Zeichnung eines grafischen Zeichens auf den Displayschirm	
SPEICHERDATEN-LESEKOMMANDOS (Memory Data Read Commands)		
RDAT	(Read Data from Display Memory) Lesen von Datenwörtern bzw. -bytes aus dem Bildwiederholspeicher	
CURD	(Cursor Address Read Command) Lesen der aktuellen Position des Cursors	
LPRD	(Light Pen Address Read Command) Lesen der Lichtstiftadresse	
DMA-STEUERKOMMANDOS (DMA Control Commands)		
DMAR	(DMA Read Request Command) Anforderung einer DMA-Lese-Übertragung aus dem Bildwiederholspeicher des GDC	
DMAW	(DMA Write Request Command) Anforderung einer DMA-Schreib-Übertragung in den Bildwiederholspeicher des GDC	

beteiligt. Die dafür notwendige Frequenz ist für Schaltkreise in MOS-Technologie viel zu hoch. Aus diesem Grund führt eine externe Hardware mittels eines schnellen Video-Schieberegisters (in diesem Konzept die 8-Bit-Schieberegister DL299) eine Parallel-/Serienwandlung der Worte des Bildwiederholspeichers in Pixel durch.

Beispielprogramm

Das auf der 2. Umschlagseite wiedergegebene PASCAL-Listing geht auf eine Idee in /2/ zurück. Es wird lediglich nur auf die mit minimalem Aufwand zu realisierende Grafik-Betriebsart des GDC Bezug genommen. Die Darstellung erfolgt dabei in der „reinen“ Grafikbetriebsart, daß heißt, auch Textzeichen und der Cursor werden als vereinbarte Grafiksymbole gezeichnet. Es werden einige Beispiele zur Programmierung des GDC in einer höheren Programmiersprache angegeben. Dies ermöglicht eine Sofortlösung für den Einsatz des GDC in der hier vorzustellenden Hardwarevariante. Es sind

- Initialisierung des GDC
- Löschen des Bildschirms
- Zeichnen von Geraden, Punkten, Kreisen und Rechtecken
- Ausschreiben von Textzeichen und Grafikcursor
- Programmierung des Zoom-Faktors und Verschiebung des Displaybereichs (Panning)

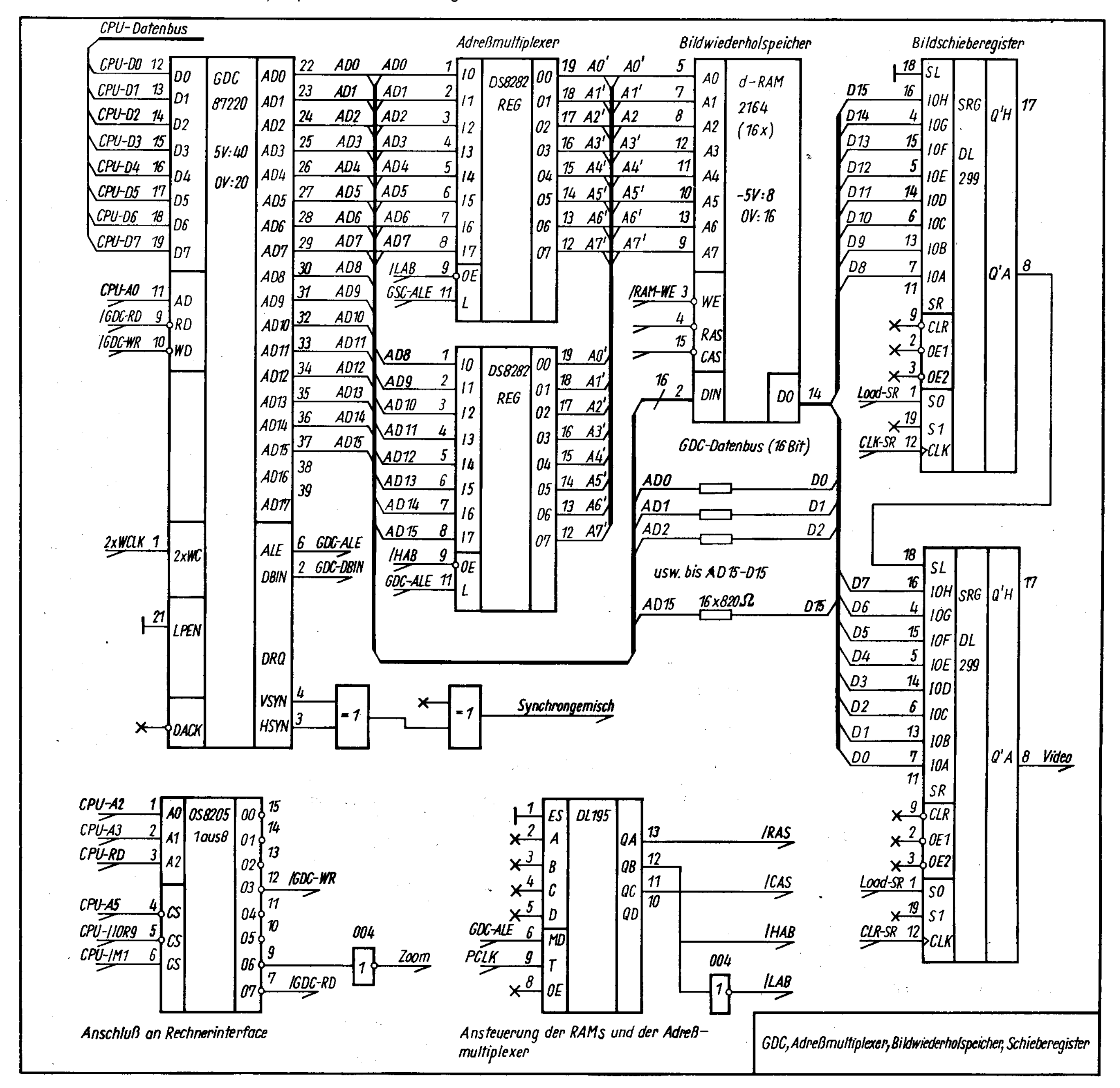
programmtechnisch implementiert. Das Vorgestellte komprimierte Programm ist ein erweiterbarer Teil eines Softwarepaketes zur grafischen Bearbeitung von Zeichnungen.

Beschreibung der Schaltung

Die zum Einsatz des U 82720 am U880-Systembus minimal notwendige Hardware ist in den folgenden Bildern 6 und 7 dargestellt. Der Bildwiederholspeicher ist aus 16 Stück 64K×1 DRAMs des Typs U2264D realisiert. Bild 5 zeigt die GDC-Interfaceschaltung und die Ansteuerung des Bildwiederholspeicher-RAMs.

Der Adreß-/Datenbus $AD_0 \ldots AD_{15}$ des GDC arbeitet im Multiplex-Verfah-

Bild 5 GDC-Interface für Bildwiederholspeicher mit Schieberegister



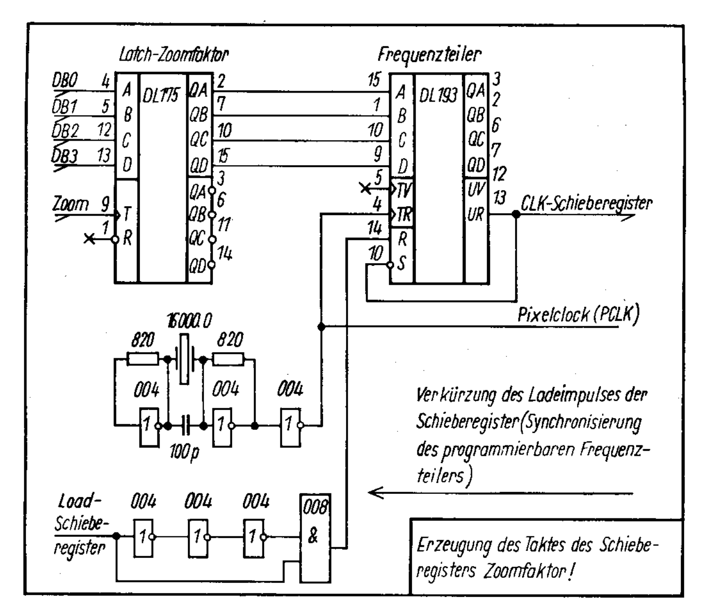


Bild 6 Erzeugung des Taktes für die Schieberegister
Bild 7 Erzeugung der Signale RAM-WE, RAM-OE und GDC-2×WCKL

Erzeugung der Signale /RAM-WE. /RAM-OE und GDC-2x Write Clock

ren. Über zwei 8-Bit-Register DS8282 werden die Adressen $A_0 \ldots A_7$ für den Bildwiederholspeicher-RAM zwischengespeichert und gemultiplext. Jeder Speicherzyklus wird mit ALE eingeleitet. (RAS = ALE). Am Schieberegister DL299 werden bezüglich $\overline{RAS}$ um jeweils 62 ns zeitverzögert die Adressenumschaltsignale $\overline{HAB}$, $\overline{LAB}$ und $\overline{CAS}$ bereitgestellt.
Sechzehn 820-Ohm-Widerstände dienen der Entkopplung des GDC-Daten-/Adreßbusses und des RAM-Datenbusses. Dadurch ist es möglich, sowohl RAMs mit gemeinsamen Datenein- und -ausgängen als auch mehrere RAM-Bildspeicherebenen (notwendig bei Farbdarstellung) anzusteuern. Die beiden 8-Bit-Schieberegister des Typs DL299 dienen der Serialisierung der Videodaten. Das zugehörige Synchronsignal wird über die GDC-Signale VSYNC und HSYNC erzeugt und in zwei Exclusiv-Oder-Gattern DL086 normgerecht gemischt. Bild 6 zeigt den Aufbau der programmierbaren Takterzeugung für die Video-Schieberegister. Die Phase dieses Schiebetaktes ist starr mit dem Signal Load-Schieberegister synchronisiert. Der Teilfaktor (Zoom-Faktor) bezüglich Maximaltakt wird im Latch DL175 gespeichert. Dieses Latch wird vom externen Steuerrechner als I/O-Port angesprochen. Über programmtechnische Beeinflussung kann hier das Verhältnis von 16 MHz Grundfrequenz (Pixelclock) zum Schiebetakt der Schieberegister programmiert werden.
Bild 7 verdeutlicht die Erzeugung der Signale Load-Schieberegister, GDC-2xWCLK, RAM-$\overline{WE}$ und RAM-$\overline{OE}$. 2xWCLK (2MHz) ist der Grundtakt des GDC, das heißt, intern arbeitet der GDC mit einer Taktfrequenz von 1 MHz. RAM-$\overline{WE}$ muß auf die Weise erzeugt werden, da der GDC kein eigenes WE-Signal erzeugt. Bei Einsatz von RAMs mit verbundenem Datenein- und -ausgang (z. B. Typ 4464, Organisation 64K×4) ist das erzeugte Signal RAM-$\overline{OE}$ anzuwenden. Die maximale Punktfolgefrequenz ergibt sich daraus, daß ein Display-Zyklus zwei Taktperioden 2xWCLK benötigt. Innerhalb dieser Zeit, die einer Frequenz von 1MHz entspricht, müssen die 16 Bit Videoinformation serialisiert werden. (1MHz×16)

Literatur
/1/ GDC U82720 – Technische Beschreibung. VEB Mikroelektronik „Karl Marx" Erfurt, 1987
/2/ Pernards, P.: CAD in Turbo-PASCAL. mc 1985 Heft 8, S. 70-78

KONTAKT
VEB Mikroelektronik „Karl Marx" Erfurt, Applikation Bauelemente, Abt. CEE, Rudolfstr. 47, Erfurt, 5010; Tel. 51076 App. 58

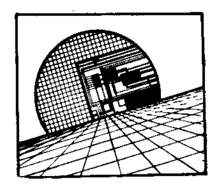
popFORTH

An der Sektion Technische Elektronik der Wilhelm-Pieck-Universität Rostock wurde eine überarbeitete Version der figFORTH-8080-Version für CP/M-kompatible Betriebssysteme mit dem Namen popFORTH geschaffen, bei der die nutzerunfreundlichen Eigenarten der figFORTH-Version behoben wurden. Die entscheidenen Charakteristika von popFORTH sind:

- Automatische Anpassung an den Speicherraum (TPA)
- automatische Anpassung an das vom Betriebssystem benutzte Disketten-Aufzeichnungsformat mit Sektoranwahl über die logische Nummer
- im Kern implementierter Editor.

Da popFORTH dem figFORTH-Standard entsprechend eine eigene physische Massenspeicherverwaltung (BIOS-orientiert) besitzt, ist es für solche Anwendungsfälle geeignet, in denen eine BDOS-orientierte Verwaltung der FORTH-Dateien nicht benötigt wird, so daß z. B. Übergang auf physische Verwaltung durch andere Betriebssysteme oder die totale Übernahme aller Betriebssystemfunktionen durch FORTH selbst möglich ist.
popFORTH ist frei zur öffentlichen Verbreitung und wird gebührenfrei abgegeben (nach Einsendung einer 40spurigen formatierten Minidiskette im SCPX-Standardformat).
Die Diskette wird bespielt mit der kompletten Assemblerquelle, dem COM-File und einer knappen Dokumentation. Falls Nutzer unbedingt auf BDOS-orientierte logische Verwaltung der FORTH-Dateien angewiesen sind, kann auf die im gleichen Haus entwickelte sehr komfortable comFORTH-Version verwiesen werden (Gesamtumfang aller Nutzerpakete und Dokumentationen z. Z. etwa 11 Minidisketten), die im Rahmen der Nachnutzung verkauft wird. *Neuthe*

KONTAKT
Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, Bereich Automatische Steuerungen, Albert-Einstein-Str. 2, Rostock 6, 2500, Tel. 4 52 83.



ISACAD – Entwurfssystem für Gate-Array-Schaltkreise

Prof. Dr. Werner Groß
Technische Universität Dresden
Sektion Informationstechnik

1. Einführung

Zur wesentlichen Beschleunigung des Entwurfs von Gate-Array-Schaltkreisen sind durchgängige anwenderfreundliche und weitgehend automatisierte Entwurfssysteme einzusetzen, die außerdem die Fehlerfreiheit des Erstentwurfs und eine hohe Ausbeute in der Fertigung sichern.
Das Entwurfssystem ISACAD realisiert diese Forderungen weitgehend, wobei die Automatisierung des Entwurfs nur mit „softwarefreundlichen“ Grundchips, den sog. Mastern, effektiv möglich wird.

2. Aufbau der neuen Grundchips des HFO-ISA-Systems

Den prinzipiellen Aufbau der digitalen ISA-Grundchips zeigt am Beispiel des ID60 Bild 1. Der Chip besteht aus 12 I2L-Transistorzellen TZ, die durch den Injektor in 24 Einzelzellen zu je 18 Transistoren getrennt werden.
Die Verbindung zwischen den Transistoren erfolgt über Kanalzellen mit den 3 Kanaltypen 1, 2, 3. Da nur eine Leitbahnebene vorgesehen ist, sind Unterführungen in den Kanalzellen für Leitungskreuzungen vorhanden. Bei der Konzeption der Grundchips wurde darauf geachtet, daß alle Konstruktionspunkte der Kanal- und Transistorzellen in einem einheitlichen Raster liegen, so daß gerade Leitbahnführungen möglich werden. Die Transistorzellen beinhalten 3 verschiedene Transistortypen, die so angeordnet sind, daß für viele Schaltungsbeispiele ebenfalls gerade Leitungsführungen möglich werden. Der Abstand zwischen zwei Transistoren wurde so gewählt, daß zusätzlich zwei Leitungen zwischen ihnen führbar sind. Parallel zum Injektor sind acht Leitbahnpositionen möglich. Die Verdrahtung erfolgt in den Kanalzellen orthogonal. In den Transistorzellen sind zusätzliche Schrägen möglich, wenn dabei die Entwurfsregeln nicht verletzt werden. Auf dem Chip sind maximal 6 verschiedene Injektorspannungen möglich, innerhalb einer Spalte (X = Konstant in Bild 1) allerdings nur zwei. Die Parallelschaltung von Basen ist zulässig, wenn dabei auch entsprechende Zusammenschaltungen von Kollektoren der vorgeschalteten Treiberstufen vorgenommen wurden.

3. Das ISACAD-System

3.1. Allgemeiner Überblick

Den Grobaufbau von ISACAD zeigt Bild 2. Nach dem Start erfolgt durch Eingabe des Nutzernamens die Zulassung zur Arbeit mit ISACAD oder die Zurückweisung. Nach Angabe des Schaltungsnamens (Objekt) und des Chiptyps (z. B. ID60) kann der Nutzer eine der angegebenen Funktionen auswählen.
Die Komplexe des Entwurfs sind:

Netzwerkeingabe (Zweig N)
Simulation (Zweige S und Q)
Layoutherstellung (Zweige, P, D, R und A).

Neben diesen angegebenen Funktionen existieren weitere, die die Arbeit mit ISACAD erleichtern:

H Hilfestellung für den Dialograhmen und die Arbeit mit Problemprogrammen
J Rücksprung in die nächst höhere Ebene
F Beenden der Arbeit.

Wird über den Dialograhmen ein Problemprogramm erreicht (z. B. mit S für Simulation und nachfolgend T für Timingsimulation), so sind weitere Funktionen möglich:

C Regiedatei editieren
P Parameterdatei editieren

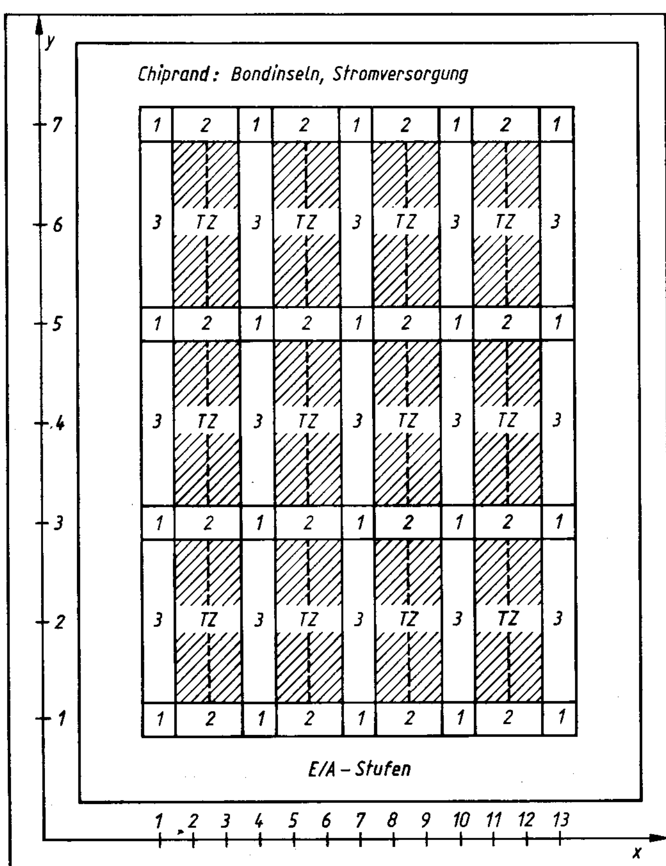


Bild 1 Aufbau des Grundchips ID60

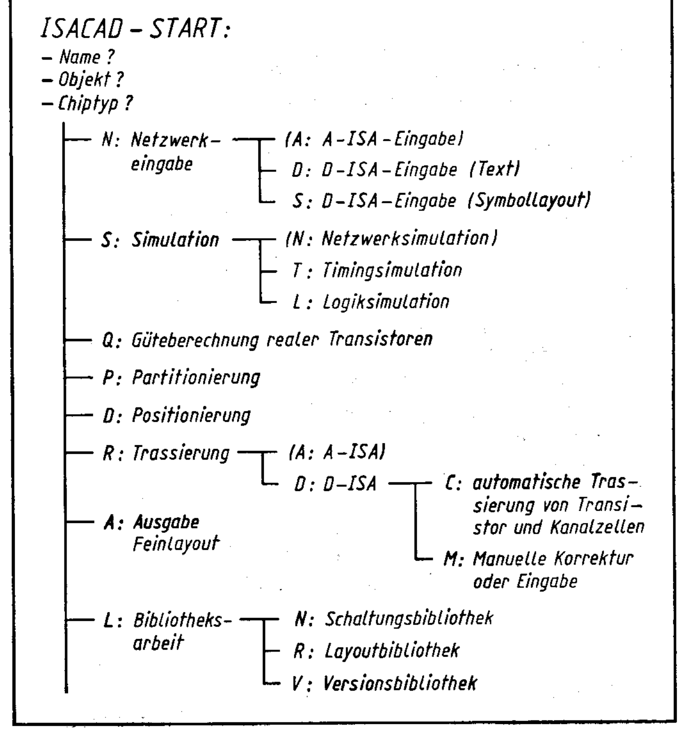


Bild 2 Grobaufbau des ISACAD-Systems

T Testfolgedatei editieren (für Simulation)
R Programmstart
W Ergebnisausgabe.

Generell ist eine Symbolkettung möglich. So bedeutet z. B. STR = Programmstart Timingsimulation aus der Startebene (Voraussetzung: Netzwerkeingabe, Regie-, Parameter- und Testfolgedatei vorhanden).
Werden bestimmte Dateien vom Nutzer nicht editiert, wird mit Standardannahmen weitergearbeitet.
ISACAD unterstützt die Versionsarbeit. In jeder Ebene sind neun verschiedene Versionen möglich. Wird in irgendeiner Ebene eine Version verändert, so werden die aus dieser Version abgeleiteten nachfolgenden Varianten unterer Ebenen im Sinne einer hohen Entwurfssicherheit gelöscht. ISACAD erlaubt die gleichzeitige Arbeit mehrerer Nutzer an der gleichen Schaltung.

3.2. Arbeit mit den Problemprogrammen

3.2.1. Netzwerkeingabe

Die Netzwerkeingabe erfolgt auf Logik- oder/und auf Elektrikniveau oder als Symbollayout.
Es können beschrieben werden:

- Transistorstrukturen
- Logik-Grundgatterstrukturen
- Boolsche Gleichungen
- Blöcke
- Felder.

Blöcke bestehen selbst aus Blöcken oder aus Elektrik- bzw. Logik-Grundgatterstrukturen, so daß die hierarchische Netzwerkeingabe möglich wird. Nach der Netzwerkeingabe erfolgen die Syntaxprüfung, die Prüfung auf Einhaltung formaler logischer und elektrischer Entwurfsregeln und die Umwandlung aller Strukturen auf logische bzw. elektrische Grundstrukturen zur Vorbereitung der Timing- bzw. Logiksimulation. Die Eingabe unterscheidet Schaltungskopf und Schaltungsrumpf. Der Schaltungskopf hat folgendes Aussehen:

SCH: Schaltungsname/Schaltungsbibliothek
[FEL:] Angabe von Laufvariablen von Feldern
[INP:] Eingangssignalnamen
[OUT:] Ausgangssignalnamen
[DEF:] Definition der Signalverknüpfungen im Schaltungsrumpf

Im folgenden werden Beispiele für den Schaltungsrumpf angegeben:

- *Transistoreingaben*

Bild 3 zeigt eine einfache Transistorschaltung. Die Notation dafür lautet:
E1,T2,4,I2 (Injektor 2)

Prof. Dr. sc. techn. Werner Groß (49) studierte von 1957 bis 1963 Schwachstromtechnik an de TH Dresden; 1968 Promotion A, 1975 Promotion B an der TU Dresden; 1971 Berufung zum Hochschuldozenten, ab 1975 Stellvertreter des Sektionsdirektors. Von 1980 bis 1983 war er als Bereichsleiter im VEB ZFTM Dresden tätig. 1985 erfolgte die Berufung zum außerordentichen Professor an der Sektion Informationstechnik der TUD, Stellvertreter des Sektionsdirektors für Forschung; 1986 Berufung zum ordentlichen Professor und zum Prorektor für Natur- und Technische Wissenschaften der TUD.

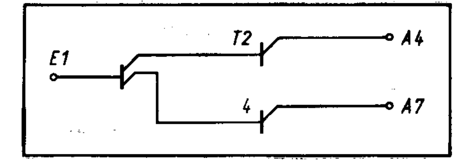


Bild 3 Transistorschaltung

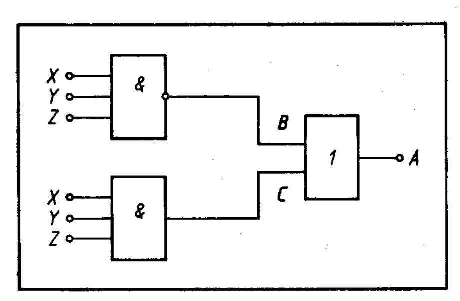


Bild 4 Logikschaltung aus Grundgattern

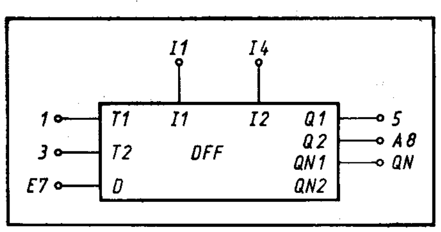


Bild 5 Transistorblock

T2,A4 (Standardinjektor 1)
4, A7, I4 (Injektor 4)

- *Logik-Elementar-Gatter-Eingaben*

Als Logikelementgatter sind NOT, OR, NOR, AND, NAND, Äquivalenz, Antivalenz, RS-Flip-Flop, D-Latch und taktflankengesteuertes D-Flip-Flop zugelassen. Für die Schaltung in Bild 4 ergibt sich folgende Schreibweise:
B=NAND (X,Y,Z)
C=AND (X,Y,Z)
A=OR (B,C).
Den Gattern werden zunächst Standardverzögerungszeiten ton und toff zugeordnet, die später in einer Bestätigungssimulation qualifiziert werden.

- *Eingabe von Boolschen Gleichungen*

Dazu werden die Zeichen + (OR), * (AND) und − (NOT) genutzt. Die Auflösung Boolscher Gleichungen erfolgt nach folgender Priorität:
() > − > * > +
Beispiel:
A = −(X*Y*Z) +X*Y*Z (siehe auch Bild 4)
Die Auflösung ergibt:
A = OR (!1, !2)
!1 = NAND (X,Y,Z)
!2 = AND (X,Y,Z)

- *Blockeingaben*

Im Beispiel Bild 5 soll der auf Transistorniveau beschriebene Block DFF

SCH: DFF
INP: T1,T2,D,I1,I2
OUT: Q1,Q2,QN1,QN2
DEF: ...

als Block mit dem Namen DFF1 in einer größeren Schaltung eingesetzt werden. Die Beschreibung dafür lautet:
DFF1(5,A8,QN,?)=
DFF(1,3,E7,I1,I4).
Das Symbol ? bezeichnet unbeschaltete Ausgänge. Unbeschaltete Eingänge werden mit EINS (elektrisch offen) oder NULL (auf Masse gelegt) gekennzeichnet.

- *Feldeingaben*

Im Beispiel Bild 6 soll eine asynchrone Teilerkette aus 10 Teilerflipflops TFF aufgebaut werden. Die Notation dafür lautet:

SCH: TEIL10
FEL: I=1:10
INP: Q'1'
OUT: Q'11'
DEF: Q'I+1'=TFF(Q'I').

Abschließend ist in Bild 7 ein komplettes Beispiel eines 5:1-Teilers angegeben.

- *Eingabe von Symbollayouts*

Bild 8 zeigt das Symbollayout einer verdrahteten Transistorzelle. Der Schaltungsrumpf dieser Zelle ist dann folgendermaßen angebbar (vereinfachte Darstellung):

DEF: ZELLE 2,3
T1,C3,*,C4,*,B1,C2,*,I1
T2,B3,*,*4,*,C1,*2,*,I1
T3,*3,*,*4,C5,C6,B2,*,I1
...

Angegeben werden die Transistornummer, der Kontaktname (C,B,I,*)

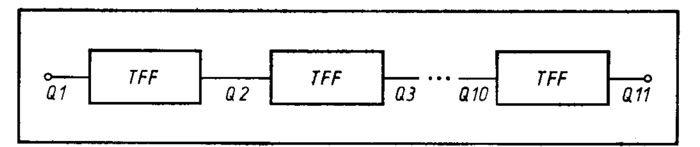


Bild 6 Teilerkette

```
Quelltext 5:1-Teiler
--------------------

SCH:    DFF1
INP:    C1,C2
OUT:    Q1,Q2,QR
DEF:    C1,102,C2,105
        102,C1
        C2,104,106,QR
        104,102,C2
        105,106,Q1,Q2
        106,104,105

SCH:    DFF2
INP:    C1,C2,IR
OUT:    Q1,Q2,QR
DEF:    C1,102,C2,105


SCH:    T510
INP:    C1,C2
OUT:    Q1,Q2
DEF:    FF1(C3,C4,?  )=DFF2(C1,C2,QR1)
        FF2(C5,C6,QR1)=DFF2(C3,C4,QR2)
        FF3(Q1,Q2,QR2)=DFF1(C5,C6)

Aufgeloeste Schaltung:
----------------------

SCH: T510
INP: C1,C2
OUT: Q1,Q2
DEF:    C1,FF1.102,C2,FF1.105   <--------\
        FF1.102,C1              <-------- \
                                           \  Namens-
                                           /  kettung
                                          /
        FF3 106,FF3 104,FF3.105 <--------/

 AUSNUTZUNG DER CHIPKAPAZITAET:      4.2%

 ANZAHL DER I2L-GATTER AM INJEKTORKNOTEN "I1":  18
                                         "I2":   0
                                         "I3":   0
                                         "I4":   0
                                         "I5":   0
                                         "I6":   0
```

Bild 7 Komplettes Beispiel der Netzwerkeingabe auf Transistorniveau

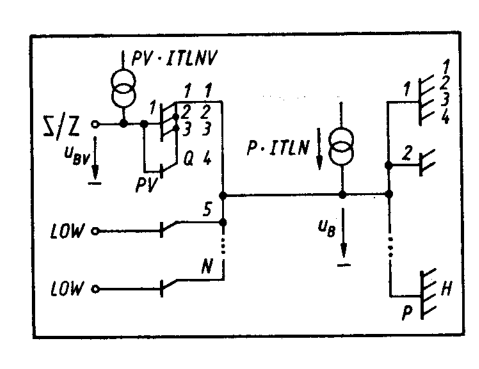


Bild 9 Transistormodell Timingsimulation

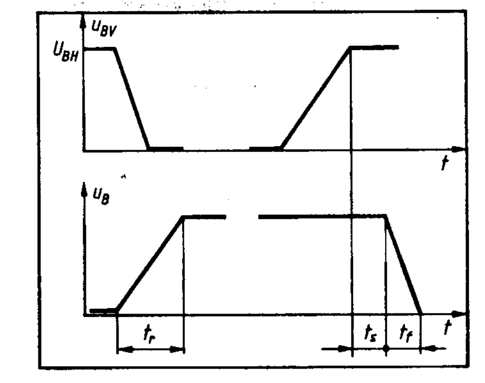


Bild 10 Zeitverhalten Timingsimulation

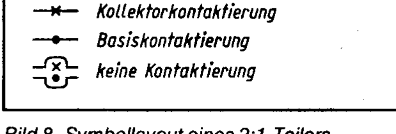


Bild 8 Symbollayout eines 2:1-Teilers

und die Leitungsnummer (falls vorhanden). Das Symbol * bedeutet, daß keine Kontaktierung besteht.

3.2.2. Simulation

Für den Nutzer unterscheiden sich die Strategien von Logik- und Timingsimulation nur wenig. In jedem Fall sind Regie- und Testfolgedateien zu erstellen, bei der Timingsimulation zusätzlich die Parameterdatei.
Beide Simulationen können als Erwartungs- und Bestätigungssimulation eingesetzt werden. Bei der Erwartungssimulation wird in der Logiksimulation noch nicht das konkrete sich aus dem Layout ergebende Zeitverhalten der Gatter berücksichtigt, bei der Timingsimulation noch nicht das Verhalten der Verbindungsleitungen. In der Bestätigungssimulation nach der Layoutherstellung wird das reale Zeitverhalten ermittelt (Timingsimulator) bzw. berücksichtigt (Logiksimulator).

- *Logiksimulation*

Das verwendete Programm DYLOS arbeitet mit äußert geringen Rechenzeiten, so daß sehr schnell für unterschiedlichste Testfolgen und Zeitverläufe Aussagen über das Verhalten der Schaltung gewonnen werden können.
Die Regiedatei enthält Anfangs- und Endzeit der Simulation, Taktzeit bei synchroner Arbeitsweise, Ausgabearten (z. B. eingeschwungener Zustand oder Einschwingverhalten), Ausgabeknoten (maximal 50) und Zeitintervall der Ausgabe. Die Testfolgedatei kann entweder durch den Initialzustand und Generatorinformationen oder durch Testbelegungen mit Zeitangaben erfolgen. Weiterhin kann im Programmablauf die Testfolge manuell über Terminal eingegeben werden, so daß der Entwerfer in Abhängigkeit der bisher erreichten Zustände seine neue Testbelegung wählen kann. Als Zustände sind zugelassen:
1=High, 0=Low, *=unbestimmt

- *Timingsimulation*

Die Timingsimulation ermöglicht die Berechnung folgender Schaltungsparameter:

- Gesamtstromaufnahme der Schaltung
- Ermittlung der Transistoren mit kritischen Gütewerten, die zu Ausbeutesenkungen führen können
- Berechnung der logischen Knotenpotentiale des Initialzustandes (Verträglichkeitssimulation)
- Berechnung des Zeitverhaltens der Schaltung mit Hilfe vereinfachter Transistormodelle.

Die Berechnung der Gesamtstromaufnahme geht davon aus, daß jeder Injektor zwei Transistoren betreibt und die Hälfte der Transistoren mit LOW belegt ist, die andere Hälfte mit HIGH.
In die Parameterdatei sind einzugeben:

- Temperatur (−25° ... +125°)
- minimal zulässige Güte (QMIN > 3!)
- Injektorstrom pro Gatter für die 6 Injektorgruppen und 1 Lastinjektor (reale Ausgangslast)
- zusätzliche Lastkapazitäten.

Zur dynamischen Simulation werden entsprechend dem Transistormodell (Bild 9) und dem Zeitverhalten der Basisknoten (Bild 10) folgende Verzögerungszeiten eingeführt:

$$t_r = (P \cdot CU + CTN + (M+N+Q)\,CD)\,\frac{UBH}{P \cdot ITLN}$$

$$t_s = 0{,}23\,\frac{1}{FTO}\sqrt{B \cdot \frac{ITLN}{ITLNV} \cdot \frac{P}{Q}}$$

$$t_f = \frac{P \cdot CU + \left(M + N - \frac{M \cdot CD}{CU + 2\,CD}\right) CD + CTN}{Q \cdot \frac{ITVMAX}{ITLN} - P - \frac{M \cdot CD}{CU + 2 \cdot CD}} \cdot \frac{UBH}{ITLN}$$

Dabei sind alle Zeiten Worst-Case-Zeiten. Da die Generatoren mit idealen Impulsen arbeiten, werden zur Simulation alle Generatoreingänge zusätzlich mit zwei Transistoren beschaltet. Bild 11 zeigt das mit dem Timingverfahren berechnete Zeitverhalten eines 5:1-Teilers.

• *Güteberechnung realer Transistoren*

Zur Ermittlung der realen Güte jedes Kollektors werden die Transistoren im Programmteil Q mit realen Basisbahnwiderständen modelliert. Als Laststrom können ein Injektorstrom verwendet werden, der dem eigenen Injektor entspricht, oder beliebige andere konstante Werte. Damit gibt Modul Q die Möglichkeit, die optimalen Injektorstrombereiche auszuwählen und auch den Übergang zwischen Transistoren unterschiedlicher Injektoren gütegünstig zu dimensionieren (Bild 12).

3.2.3. Symbollayoutherstellung

• *Partitionierung*

Die Partitionierung bildet in Abhängigkeit der Zugehörigkeit eines Transistors zu einer Injektorgruppe Unternetzwerke zu je maximal 18 Transistoren. Die Partitionierung kann die hierarchische Eingabe beachten oder die Unternetzwerke nach anderen Optimalitätskriterien erzeugen.

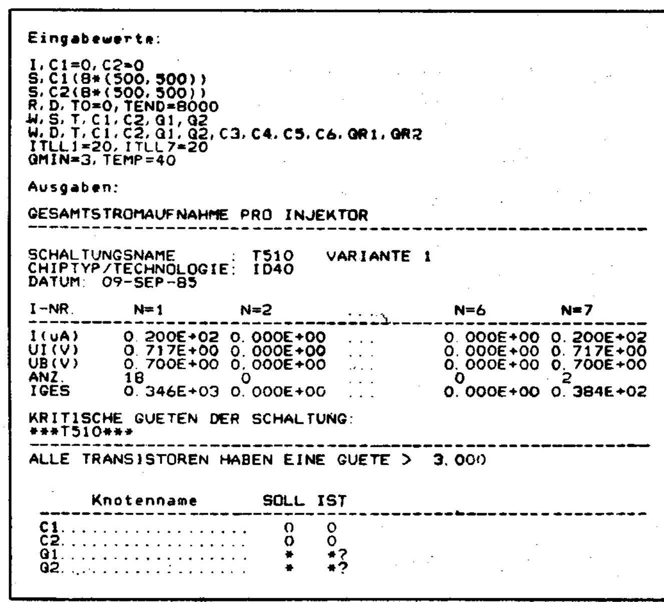
• *Positionierung*

Die Positionierung wird in 3 Etappen ausgeführt.
Etappe 1 positioniert grob die Unternetzwerke der Partitionierung entsprechend ihrer Zugehörigkeit zu Injektorgruppen auf dem Chip, wobei der Entwerfer die maximale Ausdehnung einer Injektorleitung auf zwei oder drei Zellen festlegen kann. Damit entsteht ein sogenanntes Injektorlayout.
Anschließend werden in Etappe 2 die Unternetzwerke exakt positioniert. Das Unternetzwerk mit den meisten Verbindungen zu anderen Unternetzwerken und den wenigsten E/A-Knoten wird in der Chipmitte plaziert.
Etappe 3 der Positionierung legt die Reihenfolge der Transistoren in den Unternetzwerken (Transistorzellen) so fest, daß maximale Verdrahtbarkeit in den Kanälen gesichert wird.

• *Globales Routing*

Es wird das Verbindungsnetz zwischen den Unternetzwerken mit dem Minimum an benutzten Kabelverbindungen berechnet. Die Verbindungen werden den Kanalzellen zugeordnet, wobei Verbindungen, die nicht über den kürzesten

Bild 11 a, b Zeitverhalten eines 5:1-Teilers


```
Eingabewerte:

I, C1=0, C2=0
S, C1(8*(500, 500))
S, C2(8*(500, 500))
R, D, TO=0, TEND=8000
W, S, T, C1, C2, Q1, Q2
W, D, T, C1, C2, Q1, Q2, C3, C4, C5, C6, QR1, QR2
ITLL1=20, ITLL7=20
QMIN=3, TEMP=40

Ausgaben:

GESAMTSTROMAUFNAHME PRO INJEKTOR
-----------------------------------------------------------

SCHALTUNGSNAME         : T510     VARIANTE 1
CHIPTYP/TECHNOLOGIE    : ID40
DATUM: 09-SEP-85

I-NR.       N=1          N=2       ...    N=6          N=7
-----------------------------------------------------------
I(uA)    0.200E+02  0.000E+00   ...   0.000E+00  0.200E+02
UI(V)    0.717E+00  0.000E+00   ...   0.000E+00  0.717E+00
UB(V)    0.700E+00  0.000E+00   ...   0.000E+00  0.700E+00
ANZ.     18         0           ...   0          2
IGES     0.346E+03  0.000E+00   ...   0.000E+00  0.384E+02

KRITISCHE GUETEN DER SCHALTUNG:
***T510***
-----------------------------------------------------------
ALLE TRANSISTOREN HABEN EINE GUETE >  3.000

      Knotenname          SOLL  IST
-----------------------------------------------------------
C1....................     0     0
C2....................     0     0
Q1....................     *     *?
Q2....................     *     *?
```

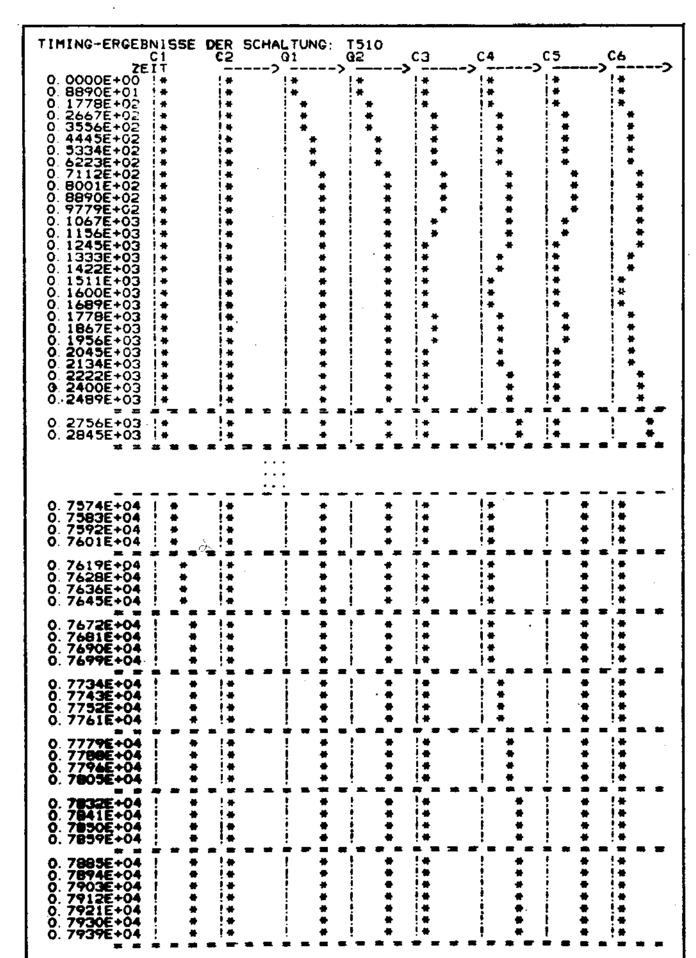
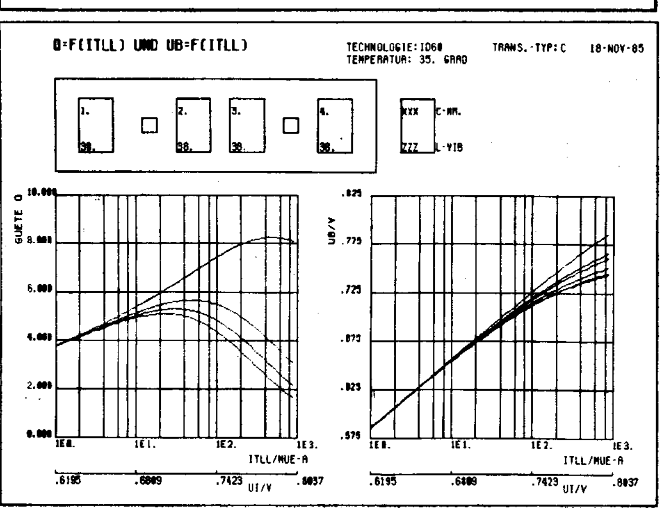
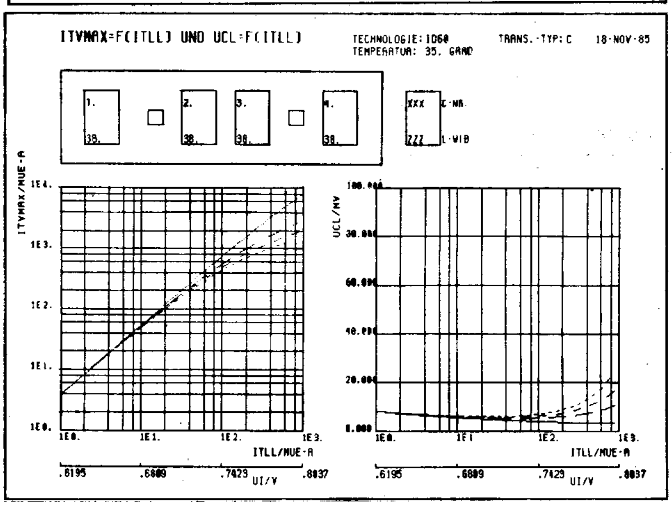


Bild 12 a, b Güteberechnung realer Transistoren





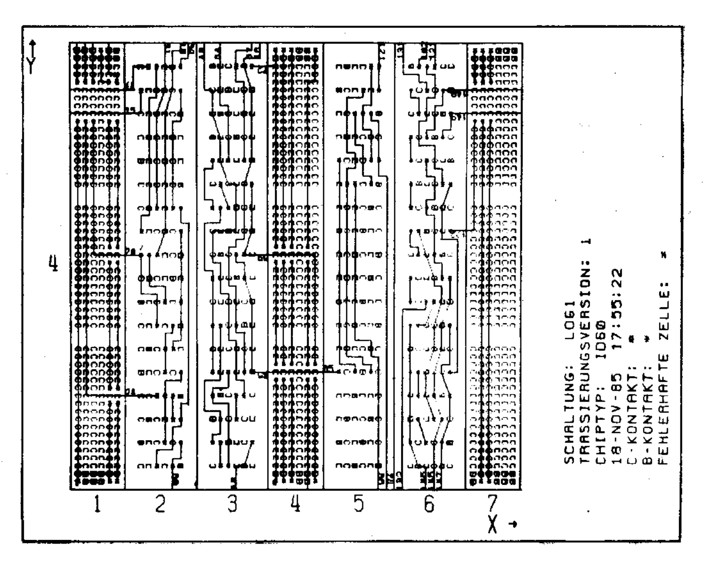


Bild 13 Symbolische Detaildarstellung

Weg geführt werden können, über Umwege geführt werden.

- *Symbollayoutgenerierung der Transistorzellen*

Auf Basis der Vorgaben des globalen Routers und der Netzwerkbeschreibung der Transistorzellen werden die Reihenfolge der Transistoren in der Zelle und das Symbollayout ermittelt.
Das Programm läßt Unterführungen durch Benutzen verschiedener Basisanschlüsse zu.

- *Symbollayoutgenerierung der Kanalzellen*

Auf Basis der Vorgaben des globalen Routers werden die Kanalzellen entsprechend dem Kanalzellentyp (1, 2 oder 3) generiert.

- *Grafische Darstellung und interaktiver Eingriff*

In symbolischer Darstellung sind sowohl Gesamtlayoutdarstellungen als auch Detaildarstellungen (Bild 13) möglich. Transistor- und Kanalzellen können durch einen Spezialgrafikeditor auf der

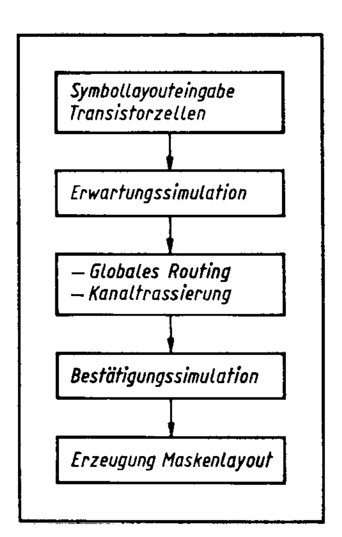


Bild 15 Entwurf mit Symbollayouteingabe

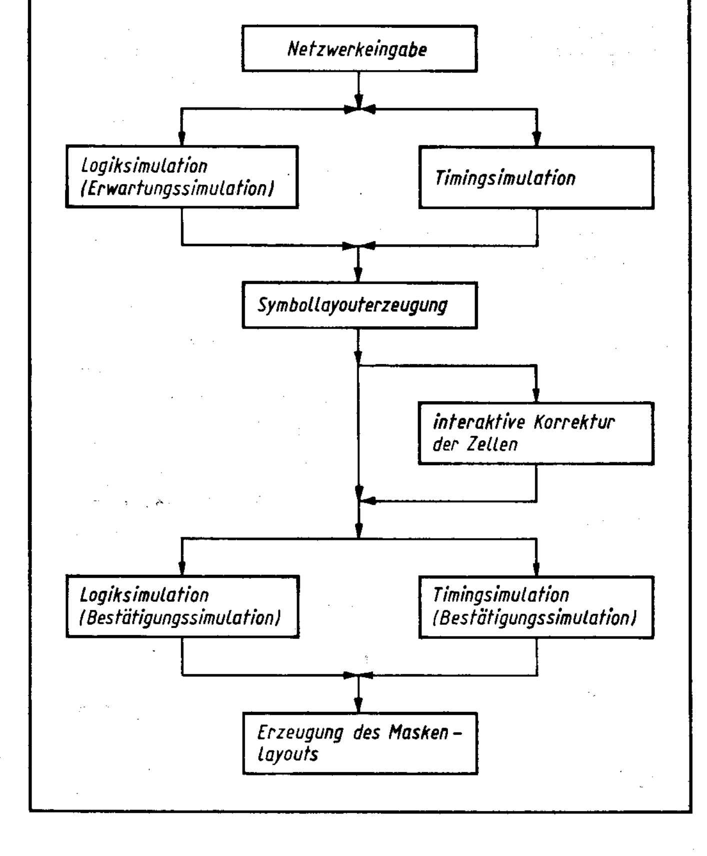


Bild 14 Top-down-Entwurfsstil

Basis der Netzwerkbeschreibung im geführten Dialog manuell erzeugt werden. Möglich ist auch der manuelle Eingriff, wenn das Programm keine Lösung findet, um anschließend maschinell weiterarbeiten zu lassen.

3.2.4. Maskenlayoutherstellung

An die I2L-Teile werden entsprechend den Wünschen des Entwerfers die Randstufen des Chips als Bibliothekselemente an das Symbollayout angefügt, so daß das Gesamtsymbollayout entsteht.
Werden diese Standard-Randschaltungen nicht verwendet, muß der Entwerfer auf der Basis des zur Verfügung stehenden Bauelementesortiments auf dem Grundchip eigene Stufen entwickeln. Aus dem erstellten Symbollayout wird das Maskenlayout automatisch generiert. Die Layoutdaten können wahlweise im Format TOPSY-80 oder GS-85 erzeugt werden.

4. Entwurfsstrategien

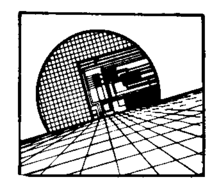
ISACAD ist für verschiedene Entwurfsstrategien flexibel einsetzbar. Bild 14 zeigt den vorzugsweise einzusetzenden Top-Down-Entwurfsstil. Zur Verbesserung der Layoutarchitektur der Zellen ist es möglich, während des Symbollayoutentwurfs Transistor- und Kanalzellen zu editieren.
Eine andere Entwurfsstrategie ergibt sich bei Eingabe des Gesamtlayouts als Symbollayoutbeschreibung aller Transistorzellen. Nach erfolgter Simulation wird über die Kanaltrassierung das Gesamtlayout erstellt (Bild 15).

5. Rechentechnische Basis

ISACAD ist auf 16-Bit-SKR-Anlagen implementierbar. Der Dialog mit ISACAD erfolgt am alphanumerischen Terminal. Die Ergebnisausgabe einschließlich Symbollayout erfolgt auf einem Plotter.

Anmerkung: Die Veröffentlichung beruht auf Arbeiten eines Jugend-Forscherkollektivs aus der Technischen Universität Dresden und des VEB Halbleiterwerk Frankfurt (Oder), dem der Autor als Betreuer angehört.

⊠ KONTAKT ☏
Technische Universität Dresden, Mommsenstr. 13, Dresden, 8027; Tel. 4630, Prof. Dr. W. Groß



16-Bit-Mikroprozessorsystem U8000

Eigenschaften und Anwendung

Dr. Heiko Kieser
VEB Mikroelektronik „Karl Marx"
Erfurt

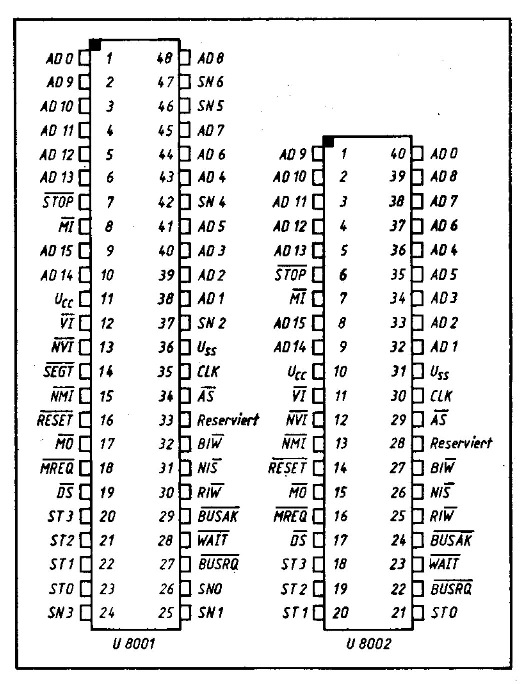
Mit den Bauelementen U8001, U8002 und U8010 aus dem VEB Mikroelektronik „Karl Marx" Erfurt steht seit einiger Zeit ein leistungsfähiges 16-Bit-Mikroprozessorsystem zur Verfügung. Das leistungsbestimmende Element des Prozessorsystems stellt die CPU dar, die in einer segmentierten und einer nichtsegmentierten Version angeboten wird. Bei der segmentierten Variante ist systemseitig der Anschluß der Speicherverwaltungseinheit U 8010 vorgesehen. Diese Speicherverwaltungseinheit (MMU – memory management unit) nimmt eine Umsetzung der logischen Adressen der CPU in physische Speicheradressen vor.
Das Mikroprozessorsystem U8000 ist als Minimalsystem konfiguriert. Damit existieren keine weiteren systemspezifischen bzw. systemeigenen Elemente. Die Leistungsfähigkeit der Peripherie wird gegenwärtig vor allem durch Komponenten aus dem System UA880 bestimmt. Aber auch andere, insbesondere hochintegrierte Peripheriecontroller mit universeller Busschnittstelle (z. B. Floppy-Disk-Controller U8272, Grafik-Display-Controller U82720) können zum Einsatz kommen.
Als wichtige Ergänzungselemente steht eine große Anzahl von sowohl MOS- als auch bipolaren Standardbauelementen zur Verfügung. In Verbindung mit weiteren Bausteinen ist diese Bauelementebasis geeignet, 16-Bit-Mikrorechner von kleinen eigenständigen Systemlösungen bis hin zu großen Mehrrechnerkonzepten zu realisieren.

Kurzcharakteristik der CPU U8000

Im Bild 1 ist die schematische Anschlußbelegung der IS U8000 dargestellt. Die CPU besitzt einen gemultiplexten 16-Bit-Daten- und Adreßbus ($AD_0 \ldots AD_{15}$), dessen aktuelle Belegung mittels der Steuersignale $\overline{AS}$ (Adress Strobe) und $\overline{DS}$ (Data Strobe) gekennzeichnet wird. Der Adressierungsbereich ist prinzipiell in Bytes organisiert (zwei Speicherbänke für gerad- und ungeradzahlige Bytes), die Datenbusbelegung mit Bytes (8 Bit) bzw. Worten (16 Bit) wird mit dem Steuersignal $B/\overline{W}$ (Byte/Word) in Verbindung mit der Adresse A_0 bestimmt. Über die Adreßlinien $A_0 \ldots A_{15}$ kann somit ein Speicherbereich von 64 KByte adressiert werden (nichtsegmentierte CPU U8002).
Dieser adressierbare Bereich wird bei der segmentierten Version U8001 mittels der Segmentleitungen $SN_0 \ldots SN_6$ auf 8 MByte erweitert. Die CPU U8001 kann auf hardwaremäßig (von der MMU) angekündigte Segmentüberschreitungen (Steuersignaleingang $\overline{SEGT}$, Segment Trap) intern reagieren. Die Aufwärtskompatibilität zwischen den U8000-CPUs ist gegeben, jedes Programm für den U8002 läuft in einem Segment des U8001.
Beide CPUs können in zwei verschiedenen Betriebsarten arbeiten (System- und Anwendermodus), die vor allem softwaremäßig hinsichtlich der Abarbeitung von privilegierten Befehlen im Systemmodus (z. B. Ein-/Ausgabeoperationen) von Bedeutung sind. Das Steuersignal $N/\overline{S}$ (Normal/System) zeigt den jeweiligen Modus an.
Über Statuslinien $ST_0 \ldots ST_3$ wird die CPU-Aktivität für jeden Zyklus kodiert angegeben. Da bei Speicheroperationen ($\overline{MREQ}$ aktiv) zwischen Befehls-, Daten- und Stackaktivitäten unterschieden wird, können hardwaremäßig sechs Speicherräume (Normal/System, Befehle/Daten/Stack) dekodiert werden. Die adressierbare Speicherkapazität läßt sich damit noch auf 384 KByte (U8002) bzw. 48 MByte (U8001) erweitern. Für Ein-/Ausgabebefehle stehen zwei Befehlsgruppen mit jeweils 16 Bit Adreßumfang zur Verfügung.
Zur Anmeldung von Interrupts sind drei Anfordereingänge vorhanden, die einem nichtmaskierbaren ($\overline{NMI}$), einem maskierbaren nichtvektorierten ($\overline{NVI}$) und einem maskierbaren vektorierten ($\overline{VI}$) Interrupt zugeordnet sind. In gleicher Weise unterstützt die U8000 intern vier verschiedene Traps (System-Aufruf, privilegierter Befehlskode im Normalmodus, erweiterter Befehl bei nichtgesetztem EPA-Bit, bei U8001: Segmentverletzung).
Weitere Steuersignale der CPU sind für Synchronisierzwecke ($\overline{WAIT}$, $\overline{STOP}$, $\overline{RESET}$), für die Bussteuerung ($\overline{BUSRQ}$, $\overline{BUSAK}$) und für die softwaremäßig unterstützte Bildung einer „daisy chain" zur Realisierung von Mehrprozessorsystemen ($\overline{MI}$ multi-micro-in, $\overline{MO}$ multi-micro-out) vorhanden.
Als Systemtakt (CLK) wird ein symmetrischer Einphasentakt mit einer Frequenz von maximal 4 MHz (UB8000-Typengruppe) verwendet. Da die CPU-Zyklen aus minimal drei Taktzuständen bestehen, läßt sich einerseits ein hoher Datendurchsatz erreichen, andererseits sind aber die erforderlichen Zugriffszeiten der Speicherbauelemente vergleichsweise zu anderen Mikroprozessoren relativ groß (400 ns).
Bild 2 zeigt die Registerstruktur der CPU U8000. Die CPU ist durch eine reguläre Architektur gekennzeichnet. Sie besitzt einen Registersatz mit sechzehn 16-Bit-Registern sowie zusätzlich einige Systemregister (Programmstatus, Zeiger zum Anfang der Interrupt/Trap-Tabelle, Refreshsteuerung). Die sechzehn allgemein nutzbaren Register können alle als Akkumulatoren verwendet und entsprechend der Darstellung im Bild 2 auch als 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Register genutzt werden. Das Register R15 (16 Bit, U8002) bzw. das Doppelregister RR14 (32 Bit, U8001) enthält den Stackpointer, der jeweils getrennt für die Anwender- und Systembetriebsart vorhanden ist.
Die Mikroprozessoren U8000 besitzen 110 verschiedene Befehlstypen, die durch Kombination mit acht Adressierungsarten insgesamt 414 leistungsfähige Einzelbefehle ergeben. Als Datentypen sind Einzelbits, zwei BCD-Digits (8 Bit), Halbbytes (4 Bit), Bytes (8 Bit), Worte (16 Bit), Doppelworte (32 Bit), Vierfachworte (64 Bit, bei Multiplikation und Division) sowie Wort- und Bytesequenzen erlaubt. Eine Sonderstellung nehmen die privilegierten Befehle ein, die aufgrund einer Schutzmaßnahme nur in der Systembetriebsart wirken. Hierdurch lassen sich beispielsweise bei der Programmierung Betriebssystemfunktionen von Anwenderprogrammen trennen.

Kurzcharakteristik der MMU U8010

Die Speicherverwaltungseinheit U8010 realisiert die Übersetzung der von der CPU ausgesendeten Adressen (logische Adressen) in Speicherplatzadressen (physische Adressen). Damit ergeben sich einerseits softwareseitig Vorteile bzgl. der Adressenunabhängigkeit der einzelnen Programme (Unterstützung des Mehrprogrammbetriebs). Andererseits kann eine Vielzahl von Schutzfunktionen beim Zugriff auf den Speicher geschaffen werden. Hierzu kann die MMU sowohl eine Meldung an die CPU absetzen (Segment Trap) als auch den eigentlichen Speicherzugriff unterbinden (Suppress-Signal).
Im Bild 3 ist die schematische Anschlußbelegung der MMU U8010 dargestellt. Ihr ist zu entnehmen, daß die Speicherverwaltungseinheit den H-Teil des



U8001

Programm-Status-Register

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	reserviertes Wort
SEG S/N EPA V/E NV/E 0 0 0 C Z S P/V DA H 0 0	Flag/Control-Wort
0 Segmentnummer 0 0 0 0 0 0 0 0	Befehlszähler
Segmentoffset	

Programm Status Area Pointer

0 Segmentnummer 0 0 0 0 0 0 0 0
oberer Offset 0 0 0 0 0 0 0 0

Refresh-Zähler

RE Rate Zeile

U8002

Programm-Status-Register

0 S/N EPA V/E NV/E 0 0 0 C Z S P/V DA H 0 0	Flag/Control-Wort
Adresse	Befehlszähler

Programm Status Area Pointer

oberer Offset 0 0 0 0 0 0 0 0

Refresh-Zähler

RE Rate Zeile

	MMU U8010	
CS 1		48 N/S
DMASYNC 2		47 R/W
SEGT 3		46 AS
SUP 4		45 DS
RESET 5		44 ST_0
A_{23} 6		43 ST_1
A_{22} 7		42 ST_2
A_{21} 8		41 ST_3
A_{20} 9		40 AD_8
A_{19} 10		39 AD_9
V_{cc} 11		38 AD_{10}
A_{18} 12		37 AD_{11}
A_{17} 13		36 C
A_{16} 14		35 V_{SS}
A_{15} 15		34 AD_{12}
A_{14} 16		33 AD_{13}
A_{13} 17		32 AD_{14}
A_{12} 18		31 AD_{15}
A_{11} 19		30 SN_0
A_{10} 20		29 SN_1
A_9 21		28 SN_2
A_8 22		27 SN_3
reserviert 23		26 SN_4
SN_6 24		25 SN_5

Bild 1 Schematische Anschlußbelegung des U8000

Bild 2 Registerstruktur des U8000

Bild 3 Schematische Anschlußbelegung des U8010

Adreß-/Datenbusses ($AD_8 \dots AD_{15}$) und die Segmentnummern zugeführt bekommt. Sie erhält ebenfalls alle die Busbelegung charakterisierenden Signale und die Statussteuersignale der CPU. Die Programmierung der MMU erfolgt unter Nutzung der speziellen Ein-/Ausgabeoperationen der CPU. Der niederwertige Adreßbus ($AD_0 \dots AD_7$) wird von der MMU nicht benötigt, da die minimale Segmentgröße auf 256 Byte festgelegt ist. Die maximale Größe eines Segments beträgt 64 KByte. Von den Segmentnummern der CPU werden von einer MMU 64 benutzt. SN_6 dient intern als Auswahlsignal, über das zwei MMUs zur Erreichung des von der CPU U8001 unterstützten Adreßraumes angesteuert werden können.

Für jedes der insgesamt 64 benutzbaren Segmente besitzt die MMU ein Segmentbeschreibungsregister. Diese Register besitzen wiederum jeweils ein 16-Bit-Basisadreßfeld, ein 8-Bit-Begrenzungsfeld und ein 8-Bit-Attributfeld. Mittels des Begrenzungsfeldes wird die Größe des betreffenden Segmentes in Schritten von 256 Byte festgelegt. Das Basisadreßfeld liefert eine Basisadresse, zu der die von der CPU ausgesendete Offsetadresse ($AD_8 \dots AD_{15}$) addiert wird. Die resultierende Adresse wird als Speicheradresse über die Linien $A_8 \dots A_{23}$ ausgesendet. Damit wird ein physischer Adreßraum von 16 MByte geschaffen. Über die acht Bits des Attributfeldes können Zugriffsbeschränkungen (Segment ist nur lesbar; Zugriff nur im Systemmodus; Zugriff nur über DMA; Zugriff nicht über DMA; Zugriff nur zum Befehlskodeholen), eine Warnung bei Zugriff auf den letzten 256-Byte-Block des Segments vereinbart und Statusinformationen (Segment wurde beschrieben; auf das Segment erfolgte ein Zugriff) zum jeweiligen Segment abgefragt werden. Darüber hinaus sind noch andere Status- und Steuerregister vorhanden, über die weitere Eigenschaften programmiert bzw. Informationen abgefragt werden können.

Bei der Einbindung der Speicherverwaltungseinheit U8010 in ein Mikrorechnersystem ist zu beachten, daß das Signal SUP eine Unterdrückung des Zugriffs bewirken soll und damit in die Zugriffselektierung des Speichers hardwaremäßig einbezogen werden muß.

Applikationshinweise

Die Einsatzgebiete des 16-Bit-Mikroprozessorsystems U8000 sind aufgrund der oben angedeuteten Leistungsfähigkeit und der großen Flexibilität sehr umfangreich. Sie reichen vom Einsatz in Einplatinenrechnern (z. B. auch für Ablösung von U880-Varianten) über eigenständige Rechnerkonfigurationen mittlerer Ausstattung bis hin zu modularen multimasterfähigen Rechnern großer Leistung.

Bei der Anpassung der 16-Bit-Mikroprozessoren U8001/8002 an die in der DDR gebräuchlichen Bussysteme der Mikrorechnersysteme K 1520 und MMS 16 sind wegen der fehlenden Signalkompatibilität Schaltungsmaßnahmen erforderlich.

Die Prozessoren U8001 und U8002 unterstützen aufgrund ihrer Architektur effizient die Implementierung zugehöriger, leistungsadäquater Software. Hierzu existieren verschiedene Betriebssysteme (sowohl echtzeitfähige als auch ein Mehrnutzersystem) und Compiler für die Anwendung moderner höherer Programmiersprachen.

Als Entwicklungsunterstützung stehen für den U8000 mehrere, unterschiedlich konfigurierte Entwicklungssysteme auf Basis des 8-Bit-Mikroprozessorsystems U880 zur Verfügung (z. B. Bürocomputer A 5120.16 des VEB Kombinat Robotron). Ausgerüstet mit dem Betriebssystem UDOS, machen sie entsprechende Cross-Software (Assembler sowie die bereits erwähnten Compiler für höhere Programmiersprachen) für den U8000 nutzbar.

Termine

Expertensysteme

WER? Wissenschaftliche Sektion Computer- und Mikroprozessortechnik des Fachverbandes Elektrotechnik und Fachausschuß Expertensysteme der WGMA der KDT
WANN? 3. und 4. Dezember 1987
WO? Erfurt
WIE? Interessenten wenden sich an den Bezirksvorstand Suhl der KDT, Straße der Opfer des Faschismus 29, Suhl, 6000, Koll. Hartung, Tel. 22112

Prof. Dr. M. Roth

Programmierung in C

(Teil IV)

Dr. Thomas Horn
Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden

Im vierten Teil dieser Artikelreihe soll nun mit der Benutzung der Struktur- und Vereinigungsvariablen sowie der Initialisierung ihrer Komponenten die Behandlung der Strukturen und Vereinigungen abgeschlossen werden. Der folgende Abschnitt wird dann das bereits im ersten Teil vermittelte Grundwissen über die Funktionen durch die Behandlung des Parameterkonzepts vertiefen und auf spezielle Probleme der Anwendung von Funktionen, wie Zeiger auf Funktionen und rekursiver Funktionsaufruf, eingehen.
Der letzte Abschnitt dieses Teils erläutert die Funktionen und die Anwendung des Makro-Präprozessors, der insbesondere für die Rationalisierung der Programmentwicklung größerer Softwaresysteme und für die Systemprogrammierung von Bedeutung ist.

7.3. Benutzung von Struktur- und Vereinigungsvariablen und deren Komponenten

In C sind mit Strukturen und Vereinigungen keine Operationen außer der Bildung von Adressen und der Adreßübergabe in Funktionen möglich.

Beispiel:
p=&a; p=t; p=&t[i];

Alle weiteren Operationen im Zusammenhang mit Strukturen und Vereinigungen beziehen sich nur auf ihre Komponenten und auf Zeiger.
Auf die einzelnen Komponenten einer Struktur oder Vereinigung kann entweder über den Variablennamen der Struktur bzw. Vereinigung oder über einen Zeiger auf die Struktur bzw. Vereinigung Bezug genommen werden.
Bei Verwendung des Variablennamens der Struktur oder Vereinigung wird der Komponentenname über einen Punkt (.) als Kettungsoperator für Komponenten angehangen.

Beispiele:
a.nr, a.pr, a.n
tlist[10].tag, tlist[10].monat,
tlist[10].jahr
tlist[i].tag, tlist[i].monat,
tlist[i].jahr

Bei Verwendung eines Struktur- bzw. Vereinigungszeigers wird über den Operator → auf die Komponenten der Struktur zugegriffen.

Beispiele:
p→nr, p→pr, p→n
tl→tag, tl→monat, tl→jahr

Die Schreibweise **p→nr** ist der Schreibweise (*p).nr äquivalent usw.
Bei der Anwendung der Inkrement- bzw. Dekrementoperatoren ist zu beachten, daß die Operatoren **.** und → eine höhere Priorität haben:

tl→tag++	Die Komponente **tag** wird inkrementiert.
++tl→tag	Die Komponente **tag** wird inkrementiert.
(tl++)→tag	Nach dem Zugriff auf die Komponente **tag** wird **tl** inkrementiert (**tl** zeigt auf die nächste Struktur).
(++tl)→tag	Vor dem Zugriff auf die Komponente **tag** wird **tl** inkrementiert.

Beispiel:
Eine Vereinigung **tab** vereinigt auf dem gleichen Speicherbereich ein Zeichenfeld **buf** mit 512 Zeichen und 64 Strukturen **art** für Artikel:
union tab {char buf [512];
struct art e [64];};

Durch die folgende Anweisung werden 10 Vereinigungen als Vereinigungsfeld **a** definiert:

union a[10], *pa;

Der Zeiger **pa** ist ein Vereinigungszeiger. Bild 4 zeigt den Aufbau der Vereinigung.
Über **a[i].buf[j]** können die Bytes (Zeichen) der Vereinigung angesprochen werden, wobei **i** die Vereinigung und **j** das Byte innerhalb der Vereinigung adressieren. Die Strukturen **art** mit ihren Komponenten können über

a[i].e[k].nr, a[i].e[k].pr und **a[i].e[k].n**

angesprochen werden, wobei **k** die Nummer der Struktur innerhalb der Vereinigung **i** angibt.

7.4. Initialisierung von Strukturen und Vereinigungen

Die Initialisierung von Strukturen ist nur für die Speicherklasse **static** bzw. für globale Strukturen möglich. Strukturen in der Speicherklasse **auto** und Vereinigungen können prinzipiell nicht initialisiert werden.
Bei der Initialisierung einer Struktur werden die verschiedenen Initialisierungswerte in einer Liste niedergeschrieben und in geschweifte Klammern eingeschlossen. Bei der Initialisierung eines Strukturfeldes sollte jedes Feldelement gesondert in geschweifte Klammern eingeschlossen werden.

Beispiele:
static struct datum a =
{22,„April“, {‚8‘, ‚5‘}};
static struct art t[]=
{„11–001“, 2.5, 10},
{„11–015“, 5.25, 120}
};

7.5. Neue Typbezeichner

Durch Anwendung der **typedef**-Anweisung können auch Strukturen- bzw. Vereinigungstypbezeichner definiert werden.

Beispiel:
typedef struct {int tag;
char *monat, jahr[2]} date;
oder
typedef struct datum date;

In beiden Anweisungen wird der neue Typbezeichner **date** vereinbart, der zur Deklaration von Variablen für die Struktur **datum** benutzt werden kann:
date a,b,c;

7.6. Rekursive Strukturen

Rekursive Strukturen sind Strukturen, die Zeiger auf eine gleichartige Struktur enthalten. Die Struktur darf sich jedoch selbst nicht enthalten, aber Zeiger sind erlaubt. Unberührt davon bleibt, daß eine Struktur beliebige andere Strukturen, die vor ihr definiert wurden, in sich enthalten darf. Die Bezugnahme auf andere Strukturen ist kein rekursives Sprachelement.

Beispiel:
Es soll eine Struktur aufgebaut werden, die es gestattet, verschiedene Ereignisse miteinander zu verketten. Die Struktur soll enthalten:
- das Datum des Ereignisses,
- die Adresse einer Zeichenkette, die das Ereignis spezifiziert,
- einen Zeiger auf das vorhergehende Ereignis und
- einen Zeiger auf das nachfolgende Ereignis.

```
struct entity {struct datum TAG;
    char *text;
    struct entity *vorher;
    struct entity *nachher;};
struct entity a,b,c,d,*i;
```

Zugriff auf die Strukturelemente der Struktur **a**:

```
a.TAG.tag
a.TAG.monat
a.TAG.jahr
a.text
a.vorher
a.nachher
```

Zugriff auf die Strukturelemente des Nachfolgers:

```
i=a.nachher
i→TAG.tag
i→TAG.monat
i→TAG.jahr
i→vorher
i→nachher
```

7.7. Bitfelder

Auf der Grundlage der Strukturen können auch Bitfelder vereinbart werden. Bitfelder sind immer vom Typ **unsigned.** Die Bitanzahl darf ein Integerwort nicht überschreiten (maximal 16 Bits). Die Bitanzahl wird nach dem Komponentennamen über Doppelpunkt getrennt spezifiziert. Für Zwischenräume können auch unbekannte Bitfelder benutzt werden. Die Bitanzahl 0 kann zur Ausrichtung auf das nächste Integerwort verwendet werden. Die Bits werden in den meisten Rechnersystemen von rechts nach links im Integerwort angeordnet.

Beispiel:
Definition der Struktur eines Steuerregisters für ein peripheres Gerät:

```
struct dk {
    unsigned go:1;
    unsigned function:3;
    unsigned adrext:2;
    unsigned ready:1;
    unsigned trap:1;
    unsigned:3;
    unsigned busy:1;
    unsigned errcode:3;
    unsigned error:1;
    } ctrldk1;
```

Auf die Bitfelder kann wie in Strukturen üblich zugegriffen werden. Bitfelder funktionieren wie kleine ganze Zahlen ohne Vorzeichen und können wie andere Werte in Ausdrücken verwendet werden.

Beispiele:

```
if (ctrldk1.error==1) printf („...“);
ctrldk1.function=2;
/* Function read */
crtldk1.go=1;
/* Start controller */
```

Hinweise:

1. Bitfelder haben keine Vorzeichen.
2. Es gibt keine (indizierten) Felder von Bitfeldern.
3. Bitfelder haben keine Adressen.

7.8. Aufzählungstypen

In neueren Implementierungen gibt es Aufzählungstypen, bei denen wie in PASCAL die möglichen Werte explizit aufgeführt werden. Den aufgezählten Werten werden intern die Integerzahlen ab 0 beginnend zugewiesen. Die Vereinbarungen der Aufzählungstypen und der Variablen für Aufzählungstypen werden ähnlich wie in Strukturen über das Schlüsselwort **enum** vereinbart.

```
enum Farbe {weiß, schwarz, rot,
gelb, grün, blau};
enum Farbe f1, f2;
```

Die Variablen **f1** und **f2** sind vom Aufzählungstyp **Farbe** und können die oben spezifizierten Werte annehmen. Mit Variablen vom Aufzählungstyp können alle Operationen ähnlich wie mit Variablen vom **int**-Typ ausgeführt werden, wie Zuweisungen, Vergleiche usw.

Beispiele:

```
f1=weiß; f2=rot;
if (f1==gelb) f1=blau;
```

Hinweis:

Aufzählungstypen werden insbesondere vorteilhaft für die Definition der Werte von Bitfeldern verwendet, was die Programmierung von Operationen mit Bitfeldern wesentlich vereinfachen kann.

8. Funktionen

Größere Problemstellungen werden mit Hilfe von Funktionen in kleinere zerlegt. Diese Vorgehensweise gestattet die Entwicklung modularer, gut strukturierter Programmsysteme. Häufig benötigte, universell nutzbare Funktionen können in Objektmodulbibliotheken katalogisiert und in andere Programmsysteme eingebunden werden. Damit wird das Programmsystem im Ganzen klarer, und eine mehrfache Entwicklung von Funktionsmoduln wird vermieden.

8.1. Definition von Funktionen

In C ist jedes Programm grundsätzlich eine Funktion, die von anderen Funktionen aufgerufen werden kann. In einer Übersetzungseinheit können mehrere Funktionen definiert werden, jedoch ist die Definition von Funktionen innerhalb von Funktionen nicht zulässig. Jede Funktion kann auch separat übersetzt werden.
Eine Funktion hat folgenden prinzipiellen Aufbau:

type name(liste_der_formalen_parameter)
deklaration_der_formalen_parameter;
{
funktionsblock
}

Der Name *name* legt den Funktionsnamen fest. In runden Klammern erfolgt die Spezifikation der formalen Parameter, falls die Funktion formale Parameter hat. Nach der Parameterliste erfolgt die Typenvereinbarung der formalen Parameter. Nach der Vereinbarung der formalen Parameter folgt der Funktionsblock, der in geschweiften Klammern Vereinbarungen und Anweisungen enthält. Die Funktion wird verlassen, das heißt, die Steuerung wird an die aufrufende Funktion zurückgegeben, wenn die rechte geschweifte Klammer erreicht wird.
Mit der **return**-Anweisung kann die Funktion vorzeitig verlassen werden. Wird in der **return**-Anweisung ein Ausdruck spezifiziert, so wird der Wert des Ausdrucks als Funktionswert an die aufrufende Funktion übermittelt. Der Typ des Funktionswertes wird durch die Typangabe *type* vor dem Funktionsnamen festgelegt. Bei ausgelassener Typangabe wird **int** angenommen.
Wird bei einer Funktion keine Speicherklasse spezifiziert, so gilt der Funktionsname als global vereinbart. Wird die Speicherklasse **static** spezifiziert, so kann auf die Funktion nur von anderen Funktionen innerhalb des Übersetzungsmoduls zugegriffen werden.
Im allgemeinen können die Funktionsnamen frei gewählt werden. Da Funktionsnamen aber in der Regel globale Symbole sind, sollten die Einschränkungen des Basisbetriebssystems berücksichtigt werden, zum Beispiel läßt das Betriebssystem OS–RW nur globale Symbole aus maximal 6 Zeichen zu, wo-

bei nicht zwischen Klein- und Großbuchstaben unterschieden wird. Weiterhin hat der Funktionsname **main** eine besondere Bedeutung; er legt die Startadresse des Programmsystems (der Task)fest. Aus diesem Grund muß die oberste Funktion, die durch das Betriebssystem gestartet werden soll, immer **main** heißen.

Beispiele:

leer(){}

Eine leere Funktion, die nichts bewirkt.

```
main()
{int i;
for(i=1;i<=10;i++)
printf(„\ni=%d j=%d\n",i,i*i);
}
```

Die Funktion kann durch das Betriebssystem gestartet werden. Sie erzeugt eine Tabelle der Quadrate der Zahlen von 1 bis 10.

```
float factor(x)
int x;
{
float y=1.0;
while (x>1) {y*=x; x-=1;}
return(y)
}
```

Funktion zur Berechnung der Fakultät der Zahl x. Der Funktionswert hat den Typ **float,** unabhängig vom Typ des Ausdrucks in der **return**-Anweisung.

8.2. Aufruf von Funktionen

Funktionen werden über ihren Namen und die Angabe einer Liste von aktuellen Argumenten in runden Klammern aufgerufen:

name(liste_der_aktuellen_argumente)

Wenn die Funktion parameterlos ist, müssen zumindest die runden Klammern angegeben werden, da der Compiler an den runden Klammern den Funktionsaufruf erkennt. Die Liste der aktuellen Argumente muß entsprechend der Liste der formalen Parameter aufgebaut sein (Stellungsparameter!). Einfache Variablen werden in C nach der Methode *„call by value"* übergeben, das heißt, es wird der Wert des entsprechenden Arguments des Funktionsaufrufes berechnet und der Variablen des formalen Parameters zugewiesen. Damit arbeitet die aufgerufene Funktion mit einer lokalen, temporären Kopie eines jeden Arguments. Das bedeutet, daß eine Funktion das ursprüngliche Argument der aufrufenden Funktion nicht verändern kann. Anders ist es bei Feldern, Strukturen, Vereinigungen und Funktionen als Argumente einer Funktion. Bei diesen Datentypen wird die Anfangsadresse des Datenobjekts übergeben, das heißt, das Datenobjekt wird nicht kopiert, sondern die aufgerufene Funktion greift unmittelbar über die übergebene Adresse auf die Datenobjekte zu. Diese Methode der Parameterübermittlung wird als *„call by reference"* bezeichnet.

Bei einem Feld werden somit die Feldelemente nicht kopiert, und es wird in der aufgerufenen Funktion keine Kopie des Feldes angelegt. Die aufgerufene Funktion kann die Feldelemente problemlos ändern. Ähnlich wird auch bei Strukturen, Vereinigungen und Funktionen verfahren. Sollen einfache Variablen in der aufrufenden Funktion verändert werden, so muß entsprechend die explizite Übergabe der Adresse der Variablen vereinbart und in der Funktion mit den Zeigern auf die Variablen gearbeitet werden.

Ein weiteres Problem ist die Typvereinbarung für den Funktionswert, der an die aufrufende Funktion übermittelt wird. Da Funktionen immer externe (bzw. **static-**) Funktionen in bezug auf die aufrufende Funktion sind, kann die aufrufende Funktion den Typ des Funktionswertes nicht kennen. Für nichtspezifizierte aufgerufene Funktionen wird standardmäßig immer **int** angenommen. Werden Werte anderer Typen übermittelt, so muß die Funktion spezifiziert werden, zum Beispiel kann die Funktion **factor** aus Pkt. 8.1. wie folgt spezifiziert werden:

float factor();

Damit ist festgelegt, daß **factor** eine (externe) Funktion ist und der Funktionswert den Typ **float** hat. Bei Funktionen, die durch den Taskbuilder eingebunden werden, kann keine Typwandlung mehr vorgenommen werden, so daß bei nicht richtiger Spezifikation des Funktionswertes schwerwiegende Fehler entstehen können.

Beispiel:

Es sei eine Struktur **datum** gegeben. In **main** sei ein Strukturenfeld **tlist** vorhanden. Es soll eine Funktion **search** definiert werden, die eine Struktur mit einem bestimmten Monatsnamen sucht und den Zeiger als Funktionswert übergibt.

```
struct datum{int tag; char*monat,
jahr[4]};
main()
{
struct datum tlist[100],
*search(),*i;
...
i=search(tlist,„April");
...
}
struct datum *search(l,m)
struct datum *l[]; char*m[];
{
int i;
for(;;l++)
for(i=0; l->monat[i]==m[i]; i++)
   if(m[i]==,\o') return(l);
}
```

8.3. Argumente aus der Kommandozeile

Im Betriebssystem installierte Tasks können mit Angabe einer Kommandozeile, bestehend aus mehreren Argumenten, aufgerufen werden, zum Beispiel

tsk arg1 arg2...

Diese Kommandozeile kann in der Funktion **main** ausgewertet werden. Zwei Argumente sind dafür in **main** zulässig:

argc – die Anzahl der Argumente in der Kommandozeile,

argv – Zeichenkettenfeld mit den Argumenten aus der Kommandozeile.

Beispiel:

```
main(argc,argv)
int argc; char *argv[];
{
...}
```

Nach der Betriebssystemkonvention ist **argv[0]** die Adresse des Namens, unter dem das Programm aufgerufen wurde (*tsk*). **argv[1]** ist die Adresse der Zeichenkette des ersten Arguments *arg1* usw. Wenn die Task ohne zusätzliche Argumente gestartet wurde, ist **argc** zumindest gleich 1.

8.4. Zeiger auf Funktionen

In C ist es möglich, einen Zeiger auf eine Funktion zu definieren, der an Funktionen übergeben oder in Tabellen gespeichert werden kann usw.

Beispiel:

Angenommen, es sei eine Funktion **swap (a,b)** definiert, die die **float**-Werte miteinander vertauscht und den größeren Wert als Funktionswert übermittelt.

Mit

float swap();

kann die Funktion deklariert werden.

Mit

float (*funct)();

kann ein Zeiger auf eine Funktion vereinbart werden, die einen Funktionswert vom Typ **float** als Resultat liefert. Die Adresse der Funktion **swap** kann wie folgt gebildet werden:
funct=(&swap)();
Eine Funktion kann über einen Zeiger wie folgt gestartet werden:
f0=(*funct)(tab[i], tab[i+1]);

Wird eine Funktion als Parameter einer anderen Funktion übergeben, so wird stets die Adresse der Funktion übergeben. Der Adreßoperator ist in diesem Fall unnötig.

Beispiel:

```
main()
{
float swap(), sort();
...
sort (a,b,swap)
...
}
float sort (x,y,change)
float x[],y[],(*change)();
{
...
(*change)(x[i],y[j]);
...
}
```

8.5. Rekursiver Funktionsaufruf

In C können Funktionen rekursiv benutzt werden, d. h. daß eine Funktion sich selbst entweder direkt oder indirekt aufrufen kann.

Beispiel:
Berechnung der Fakultät einer Zahl x:

```
float factor(x)
int x
{
if (x==1) return (1.0);
   else return (x*factor(x-1));
}
```

Rekursionen sparen im allgemeinen keinen Speicherplatz, da im Stack eine Reihe von Zwischenresultaten und Adressen abgelegt werden muß. Der Stack wird dadurch sogar erheblich belastet. Ebenso sind Rekursionen nicht schneller, aber rekursive Lösungen sind in Abhängigkeit vom Problem (rekursiv definierte Datenstrukturen usw.) oft kompakter, leichter zu schreiben und zu verstehen.

Hinweis:
In Abhängigkeit vom Grad der Nutzung von rekursiven Funktionsaufrufen (Rekursionstiefe) ist der Stackbereich zu vergrößern (z. B. im Betriebssystem OS–RW mit dem Taskbuilder TKB).

9. Der Makro-Präprozessor

Der C-Compiler verfügt über einen Makro-Präprozessor, der einige Spracherweiterungen wie Fileeinfügungen, Makrosubstitutionen und bedingte Generierungsanweisungen realisiert.
Der Makropräprozessor generiert in einem Vorpaß das eigentliche Quellprogramm, das anschließend übersetzt wird. Die Anweisungen an den Makro-Präprozessor sind gegenüber den C-Sprachanweisungen zeilengebunden und beginnen mit einem Doppelkreuz (#) in der Position 1. Sie können an beliebiger Stelle im Quelltext auftreten, unabhängig von der Quellprogrammstruktur. Ihre Syntax ist nicht mit den Sprachelementen von C gekoppelt. Die Anweisungen an den Makro-Präprozessor belegen grundsätzlich immer eine ganze Zeile. Falls aber eine Zeile, zum Beispiel bei Makrodefinitionen, nicht ausreichend ist, können Fortsetzungszeilen verwendet werden, wenn die vorhergehende Zeile als letztes Zeichen einen Backslash (\) enthält.

9.1. Fileeinfügungen

#include "*filename***"**
fügt ein File aus dem aktuellen Fileverzeichnis des Nutzers in das C-Programm ein, während

#include<*filename*>

ein File aus dem Bibliotheksfileverzeichnis (im OS–RW: LB:[1,1]) einfügt. Es kann ein beliebiger C-Text eingefügt werden, insbesondere Strukturbeschreibungen, Makrodefinitionen usw. Die **#include**-Anweisung kann geschachtelt angewendet werden, das heißt, in einem eingefügten File dürfen weitere **#include**-Anweisungen benutzt werden.

9.2. Makrodefinitionen und -substitutionen

#define *identifier string*
ist eine Makrodefinition ohne Argumente und dient der einfachen Zeichenkettensetzung; zum Beispiel ersetzt

#define EOF (−1)

im weiteren C-Text alle Zeichenketten **EOF** durch **(−1).**

#define *identifier(identifier, . . .) string*

ist eine Makrodefinition mit Parametern. Bei Makroaufruf werden im Makrokörper *string* alle formalen Parameter durch aktuelle Parameter ersetzt.

Beispiel:

```
#define min (x,y) ((x)<(y)? (x):(y))
...
Amin=min(A[i],A[i+1]);
```

Infolge der Makrogenerierung wird folgender C-Text übersetzt:

```
Amin=((A[i])(A[i+1])?
(A[i]):(A[i+1]));
```

9.3. Streichen von Makrodefinitionen

#undef *identifier*
streicht die angegebene Makrodefinition aus dem Verzeichnis des Makro-Präprozessors. Damit ist der Identifikator im weiteren undefiniert.

Beispiel:

```
#undef min
#undef EOF
```

Die Makrodefinitionen **min** und **EOF** werden gestrichen.

9.4. Bedingte Compilierung

Ein bedingter Anweisungsblock wird durch eine der folgenden drei Anweisungen eingeleitet:

#if *constant-expression*
#ifdef *identifier*
#ifndef *identifier*

Die nachfolgenden C-Anweisungen werden generiert, wenn die Bedingung **TRUE** bzw. der Konstantenausdruck ungleich 0 ist. Abgeschlossen wird der bedingte Anweisungsblock durch eine Anweisung

#endif

Ist zwischen der **#if**-Anweisung und der **#endif**-Anweisung eine Anweisung

#else

enthalten, so wird die Generierungsbedingung umgekehrt. Damit werden entweder die Anweisungen bis zur **#else**-Anweisung oder danach generiert. Bedingte Anweisungsblöcke können ineinander geschachtelt werden.

Die Reihe zu C wird im nächsten Heft fortgesetzt mit Ausführungen zu den Ein- und Ausgabefunktionen und zur Methodik der Progammentwicklung.

Programmierung von Ablaufsteuerungen

Zum IEC-Standard für SPS

Dr. Werner Kunke, Steffen Zeidler
Technische Universität Karl-Marx-Stadt

Tendenzen in der Softwareentwicklung

Speicherprogrammierbare Steuerungen (SPS) zeichnen sich gegenwärtig durch zunehmende Leistungsfähigkeit und relativ niedrige Kosten aus. Ein breites Spektrum verschiedener Steuerungen führt dazu, daß in vielen Automatisierungslösungen von der Entwicklung spezieller Steuerungshardware abgesehen und statt dessen eine industriell gefertigte SPS eingesetzt wird. Damit verschiebt sich der Schwerpunkt der notwendigen Entwicklungsarbeiten von der Hard- zur Software. In der Praxis der Softwareentwicklung für SPS sind einige Besonderheiten gegenüber der Entwicklung von Anwendersoftware für Rechnerarbeitsplätze oder auch für Großrechner zu verzeichnen:

■ Die Programmierung der SPS ist nur ein Teil der für eine Automatisierungslösung notwendigen Entwicklungsarbeiten. In vielen Fällen handelt es sich um Rationalisierungsaufgaben, die von relativ kleinen, nicht auf solche Arbeiten spezialisierten Kollektiven gelöst werden müssen. Deshalb werden sehr häufig keine versierten Softwarespezialisten zur Verfügung stehen.

■ Der Test des fertigen SPS-Programms erfolgt in der bisherigen Praxis zum größten Teil auf der bereits mit dem Steuerobjekt verbundenen Steuerung. Das ergibt sich aus der Tatsache, daß das Steuerprogramm und das Steuerobjekt eine funktionale Einheit bilden. Deshalb ist es erforderlich, daß das Steuerobjekt mit angeschlossener Steuerung real zur Verfügung steht. Wenn diese Voraussetzung erfüllt ist, ist der Inbetriebnahmetermin in der Regel herangerückt, und die für den Test und die daraus resultierenden notwendigen Programm- und Dokumentationsänderungen zur Verfügung stehende Zeit wird äußerst knapp.

■ Die gegenwärtige SPS-Programmierung ist geprägt durch den ursprünglichen Anwendungsfall der Ablösung von Relaissteuerungen. Mit der heutigen Leistungsfähigkeit der SPS auf der Basis moderner mikroelektronischer Schaltkreise werden auch bisher der Hardware zugeordnete Aufgaben wie Regelungen und Positionierungen zunehmend durch Software ersetzt. Außerdem entstehen neue Anforderungen, die sich nur im Rechnerverbund lösen lassen. Die Komplexität der damit entstehenden Software erfordert auch in der SPS-Programmierung eine Technologie der Softwareerstellung, wie sie sich in der kommerziellen Datenverarbeitung bereits längere Zeit durchsetzt.

■ Es gibt eine Vielfalt unterschiedlicher Gerätetechnik, aber keine einheitlichen Softwarewerkzeuge, wie sie die höheren Programmiersprachen in der kommerziellen EDV-Anwendung darstellen.
Aus diesen Feststellungen resultiert, daß es unumgänglich ist, für die SPS-Programmierung neue und einheitliche Softwarewerkzeuge bereitzustellen. Werkzeuge, die es gestatten, in relativ kurzer Zeit sichere Softwarelösungen für komplizierte SPS-Anwendungen zu realisieren.

Die für die SPS-Programmierung zu verwendenden Sprachen sollen auf den unterschiedlichsten Steuerungstypen zur Verfügung stehen, um eine einheitliche Programmierung – unabhängig von den Besonderheiten der jeweiligen Hardware – zu unterstützen. Sie müssen internationalen Normen entsprechen, um den Export der mit SPS ausgerüsteten Maschinen und Anlagen zu erleichtern.

IEC-Standard zur Programmierung von Ablaufsteuerungen

Im folgenden wird das Konzept des IEC-Standards für Programmiersprachen von SPS entsprechend dem Dokument 65A (Secretariat) 49 vom Januar 1985 dargestellt. Es sei darauf verwiesen, daß in diesem Standard nur die „reine" SPS-Programmierung behandelt wird. Die vorgeschriebenen sprachlichen Mittel sind nicht ausreichend, wenn die SPS als integrierter Bestandteil eines komplexen Automatisierungssystems betrieben wird. Der IEC-Standard unterscheidet zwei Sprachniveaus:
Auf der unteren Ebene sind als grafisch orientierte Programmiersprachen Kontaktplan und Logikplan, als textorientierte Sprachen Befehlsliste und *Mathematische Ausdrücke* vorgesehen. Erweiterungen des vorgeschlagenen Sprachumfangs sind auf dieser Ebene durch Funktionsbausteine möglich. Für diese ist die genormte Darstellungsform einzuhalten. Alle diese Sprachen lassen sich formal ineinander überführen.

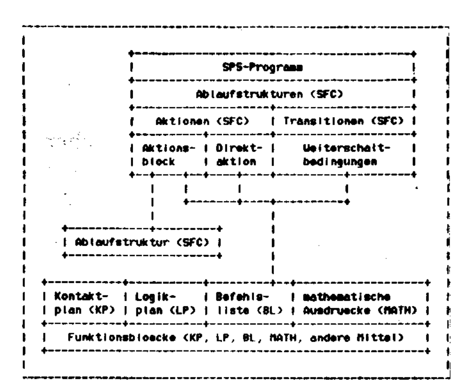


Bild 1 Hierarchie der SPS-Programmiersprachen (IEC-Standard)

Die obere Ebene dient zur Darstellung von Abläufen mit der grafischen Programmiersprache SFC (Sequential Funktion Chart). Eine äquivalente textorientierte Programmiersprache wird zur Zeit bearbeitet. Die Elemente dieser Sprache sind Schritte mit zugeordneten Aktionen und Transitionen mit zugeordneten Bedingungen. Die Aktionen und Bedingungen werden mit den Mitteln der unteren Ebene beschrieben.
Die Aktionen können aber auch selbst wieder mit SFC beschriebene Abläufe sein. Dieser Sachverhalt ist in Bild 1 als Übersicht dargestellt.

Ein Schritt kann aktiv oder passiv sein. Die den Schritten zugeordneten Aktionen sind in den häufigsten Fällen Boolesche Gleichungen, die Verknüpfungen von Eingangs- und Merkervariablen auf Ausgangs- oder Merkervariablen abbilden. Diese Verknüpfungen sind so lange wirksam, wie der Schritt aktiv ist.
Mit Schritten und Transitionen können außer den üblichen Strukturen Ablauf und Fallauswahl auch Parallelitäten dargestellt werden. Die Abläufe werden mit SFC immer vertikal notiert. Rückwärtssprünge sind nur mit Konnektoren realisierbar. In einem Ablauf werden abwechselnd Transitionen und Schritte notiert. Zwei unmittelbar aufeinander folgende Transitionen bzw. Schritte sind nicht zulässig. Die Schritte werden damit bezüglich einer Transition in Vorgänger- und Nachfolgerschritte unterteilt.
Eine Transition kann mehrere Vorgänger- bzw. Nachfolgerschritte haben. Grafisch wird das mit dem ISO-646-Zeichensatz, wie im Bild 2 gezeigt, dargestellt. Sind alle Vorgängerschritte einer Transition aktiv und alle ihre Nachfolgerschritte passiv, so „schaltet" die Transition, *falls die ihr zugeordnete Bedingung aktiv ist.* Das bedeutet, ihre Vorgängerschritte werden passiv und ihre Nachfolgerschritte aktiv. Die den Trasitionen zugeordneten Bedingungen werden als logische Ausdrücke (Boolesche Verknüpfung von Merker- und Eingangsvariablen) notiert. Ist einer Transition keine Bedingung zugeordnet, so ist das mit einer immer erfüllten Bedingung gleichwertig. Die in Bild 2

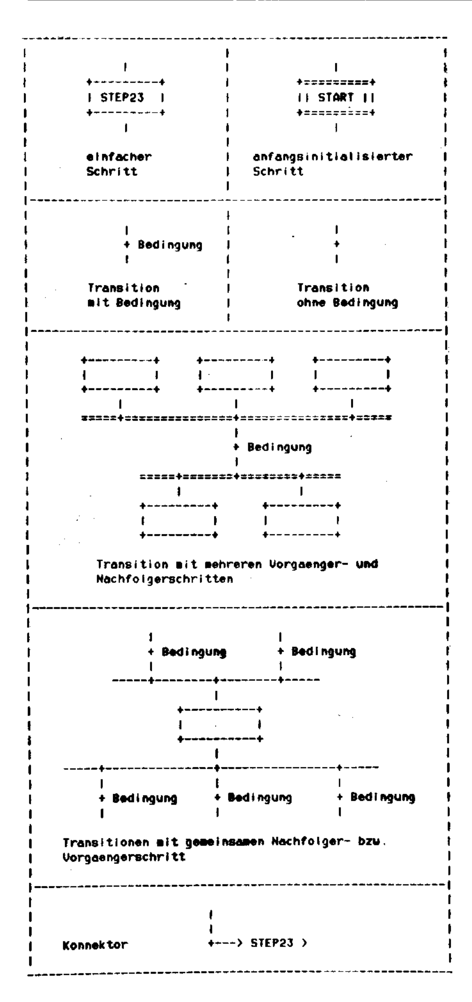


Bild 2 Darstellung der grafischen Elemente von SFC im ISO-646-Zeichensatz

gezeigten Verzweigungen sind wie folgt zu interpretieren:

Mehrere Transitionen mit gemeinsamem Vorgängerschritt stellen eine Fallauswahl dar. Es wird genau einer der folgenden Abläufe gestartet.

Mehrere Transitionen mit einem gemeinsamen Nachfolgerschritt dienen der Zusammenführung alternativer Abläufe.

Eine Transition mit mehreren Nachfolgerschritten startet mehrere Abläufe gleichzeitig (Eröffnung einer Parallelität).

Eine Transition mit mehreren Vorgängerschritten sychronisiert mehrere parallele Abläufe. Sie kann nur schalten, wenn die letzten Schritte aller an dieser Transition zusammengeführten Abläufe aktiv sind.

Eingaenge	
TEMP	Dieser Variablen ist die aktuelle Temperatur der Faerbeflotte zugeordnet.
LEER	ist TRUE falls zuwenig Flotte in der Maschine
PUMPE_EIN	ist TRUE wenn die Flotte umgewaelzt wird.
Ausgaenge	
PUMPE_H	schaltet den Hauptschuetz der Umwaelzpumpe ein
PUMPE_S	schaltet die Umwaelzpumpe auf Sternschaltung.
PUMPE_D	schaltet die Umwaelzpumpe auf Dreieckschaltung
V_DAMPF	oeffnet das Dampfventil (Heizen mit Dampf)
V_KONDENS	oeffnet das Kondenswasserventil.
V_RKUEHL	oeffnet das Kuehlwasserabflussventil
HUPE	schaltet die Hupe ein
Variablen aus dem uebergeordneten Programm	
HEIZEN_EIN	ist solange gesetzt, wie geheizt werden soll.
GRAD	gibt die maximale Temperatursteigerung pro Minute an. Falls GRAD gleich Null ist, wird mit maximaler Heizleistung geheizt.
SOLL	gibt die zu erreichende Endtemperatur an.
programminterne Variablen	
ALARM	ist TRUE falls ein unzulaessiger Prozesszustand erkannt wurde.
WARTEN	Hilfssignal fuer Zeitverzoegerung
ZWERT	Interner Sollwert fuer Temperaturregelung.
HEIZEN	Hilfssignal, TRUE falls alle Bedingungen fuer das Heizen erfuellt sind.
MIN	Am Ende des Programms mindestens zu erreichende Temperatur

Bild 3 In der Heizungssteuerung verwendete Variablen

Anwendungsbeispiel

Am Beispiel der vereinfachten Heizung einer Färbemaschine wird im folgenden ein dem IEC-Vorschlag entsprechendes Steuerprogramm demonstriert. Die Heizungssteuerung ist als Aktionsblock realisiert, der einem Schritt eines übergeordneten Steuerprogramms zugeordnet wird.

Eingänge der Steuerung sind die Ist-Temperatur der Färbeflotte und zwei Signale zur Überwachung der Umwälzpumpe und des Füllstandes der Flotte. Ausgänge sind binäre Signale zur Steuerung der Umwälzpumpe (Hauptschütz, Dreieckschütz, Sternschütz), der Ventile (Dampfventil, Ablaufventil für Kühlwasser, Kondenswasserventil) und einer Hupe.

Vom übergeordneten Programm werden der Sollwert für die Endtemperatur und eine Vorgabe für den Gradienten des Temperaturanstiegs sowie ein binäres Signal zum Beenden des Heizvorganges übernommen.

Die Zusammenstellung aller Variablen des Steuerprogramms ist Bild 3 zu entnehmen.

Folgende Steueraufgabe ist zu realisieren:

1. Falls zu Beginn des Heizvorganges die Umwälzpumpe nicht läuft, ist sie einzuschalten. Dazu sind der Hauptschütz und der Sternschütz einzuschalten. Nach sieben Sekunden wird vom Sternschütz auf den Dreieckschütz umgeschaltet. Nur bei laufender Umwälzpumpe darf der eigentliche Heizvorgang gestartet werden.
2. Zu Beginn des Heizens ist das Kühlwasserrückflußventil für eine Minute zu öffnen. Danach ist das Kondenswasserventil zu öffnen.
3. Falls ein Gradient vorgegeben wurde, ist ein Momentansollwert für die Heiztemperatur, ausgehend von der Starttemperatur bis zum Erreichen des eigentlichen Sollwertes, jeweils nach einer Minute um den Gradienten zu erhöhen.
4. Die Temperatur ist nach Erreichen des (Momentan-) Sollwertes mit einem einfachen Zweipunktregler konstant zu halten. Geheizt wird durch Öffnen des Dampfventils.
5. Der Heizmodul ist zu beenden, wenn die Bitvariable HEIZEN-EIN zurückgesetzt wird. Dabei ist die Endtemperatur zu kontrollieren. Falls die Endtemperatur kleiner als der um 5 verringerte Sollwert ist, muß für eine Minute die Hupe eingeschaltet werden.
6. Während des Heizens sind der Flottenstand und die Umwälzpumpe zu überwachen. Im Fehlerfall ist ebenfalls die Hupe zu betätigen und der Heizmodul zu verlassen.

Die dafür notwendige Ablaufstruktur in SFC-Notation ist im Bild 4 dargestellt. Diese Struktur enthält keinen anfangsinitialisierten Schritt, da sie als Aktionsblock in einem übergeordneten SFC-Ablauf enthalten ist. Beim Aufruf des Aktionsblockes wird der Schritt H_EIN aktiviert.

Im Schritt UEBERWACH werden die im 6. Punkt der Aufgabenstellung angegebene Ausnahmesituation (Signal ALARM) und die Erfüllung der Heizbedingung (Signal HEIZEN) ermittelt. Nach dem Schritt VORBEREIT wird alternativ der eingentliche Heizvorgang gestartet oder die Umwälzpumpe eingeschaltet (Punkt 1 der Aufgabenstellung).

Der Ablauf VENT_AUF, VENT_UMSCH realisiert Punkt 2 der Aufgabenstellung, die Abläufe HEIZ_AKT, HEIZ_PASS und STRT_GRAD, PASSIV realisieren die Punkte 3 und 4.

Diese drei Abläufe werden parallel aktiviert und abgearbeitet. Die Transition vor dem Schritt VENTIL_ZU synchronisiert vier parallele Abläufe und beendet den Aktionsblock.

In Bild 5 sind die den Schritten zugeordneten Aktionen aufgelistet.

Eine Besonderheit stellen die Anweisungen der Form

WARTEN = TRUE FOR zeitwert

dar. Die links stehende Bitvariable wird in diesem Fall gesetzt und unabhängig vom Zustand des Schrittes nach Ablauf der vorgegebenen Zeit zurückgesetzt. Im Gegensatz zu den anderen Anweisungen werden diese Anweisungen nur jeweils einmal zum Zeitpunkt der Aktivierung des Schrittes ausgeführt. Diese Notation ist an das Beschreibungsmittel SN700 angelehnt, da im vorliegenden IEC-Dokument die Operationen zur Zeitbeeinflussung noch nicht spezifiziert sind.

SN700-Software

An der TU Karl-Marx-Stadt wurde unter dem Namen SN700 (Steuernetze im System 700) eine Fachsprache entwikkelt, die es ermöglicht, entsprechend dem IEC-Standard entworfene Ablaufstrukturen formal zu notieren. Dazu wird (in der jetzigen Ausbaustufe) die Menge der Transitionen mit den zugehörigen Vorgänger- und Nachfolgerschritten textorientiert notiert. In zwei weiteren Programmblöcken sind die

verwendeten Variablen zu deklarieren und den Schritten die Aktionen zuzuordnen.
Es ist möglich, Aktionsblöcke als separate Quelltexte zu notieren.
Das zugehörige Programmpaket besteht aus einem Übersetzer, der die Einhaltung der Syntax überprüft und einen Zwischencode erzeugt.
Ein Verbinder vereinigt die Zwischencodes separat notierter und übersetzter Programmteile (Aktionsblöcke).
Ein Codegenerator erzeugt aus dem Zwischencode ein vollständiges SPS-Programm in Form einer Befehlsliste (MPSS, beschrieben in /1/). Damit wird eine Portabilität der Software auf verschiedene Zielsteuerungen gewährleistet.
Ein Simulator gestattet es, die Abläufe auf der Basis des Zwischencodes ohne Zuhilfenahme der Steuerung zu untersuchen. Damit wird es möglich, eine große Zahl von Fehlern bereits auf dem Entwicklungssystem zu entdecken und zu beseitigen. Eine dialogorientierte Bedienerführung erleichtert dem ungeübten Entwickler den Umgang mit dem Softwarepaket ohne spezielle Systemkenntnisse.
Die Inbetriebnahme wird in der jetzigen

```
Schritt        zugeordnete Aktion

VORBEREIT      keine
UEBERWACH      ALARM = LEER OR (NOT PUMPE_EIN AND PUMPE_D);
               HEIZEN = HEIZEN_EIN AND (TEMP<SOLL) AND
                        ((TEMP<ZWERT) OR (GRAD=0));
KONTROLLE      MIN = SOLL - 5;
HUPEN          HUPE = TRUE FOR 1m;
HEIZ_ENDE      HEIZEN = FALSE;
STRT_PUMP      PUMPE_H = TRUE;
               PUMPE_S = TRUE FOR 7m;
               PUMPE_D = FALSE;
ENDE_PUMP      PUMPE_S = FALSE;
               PUMPE_D = TRUE;
VENT_AUF       V_RKUEHL = TRUE FOR 1m;
VEN_UMSCH      V_KONDENS = TRUE;
               V_RKUEHL = FALSE;
HEIZ_AKT       V_DAMPF = TRUE;
HEIZ_PASS      V_DAMPF = FALSE;
STRT_GRAD      ZWERT = TEMP;
INCR_GRAD      ZWERT = ZWERT + GRAD;
PASSIV         WARTEN = TRUE FOR 1m;
VENTIL_ZU      V_KONDENS = FALSE
               ALARM = FALSE
```

Bild 5 Den Schritten zugeordnete Aktionen; mathematische Schreibweise entsprechend IEC-Standard

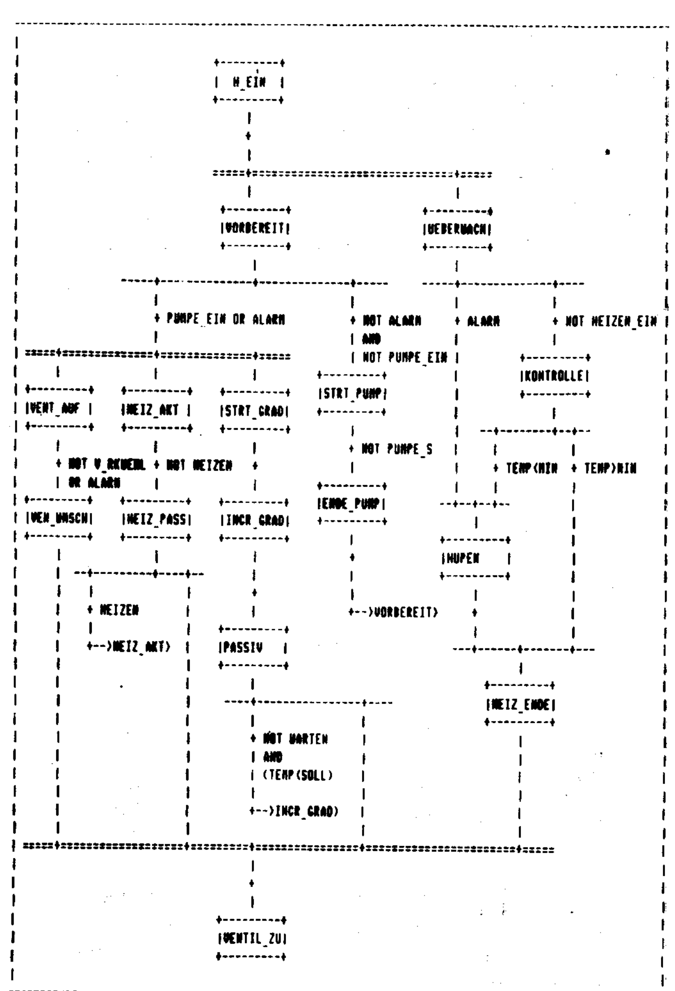
Version durch eine Referenzliste unterstützt. An Hand dieser Liste kann im MPSS-Inbetriebnahmesystem die aktuelle Belegung aller im SN700-Programm verwendeten Variablen und der Zustand aller Schritte (aktiv oder passiv) überwacht werden.
Das Softwarepaket ist auf PRG700, BC A 5120/A 5130 und PC 1715 unter dem Betriebssystem UDOS lauffähig.
Als Weiterentwicklung ist die Erstellung eines speziellen Editors für Inbetriebnahmebilder vorgesehen. Diese Inbetriebnahmebilder ermöglichen es, Variablenbelegungen und Schrittzustände in der Steuerung übersichtlich auf dem Bildschirm des Programmiergerätes anzuzeigen. Ferner befindet sich eine neue Version des Programmpaketes in Arbeit. Sie wird sich vor allem durch geringere Laufzeiten der Entwicklungssoftware auszeichnen.
Die Sprachversion wird durch Elemente erweitert, die es gestatten, Abläufe auch in textorientierter Form übersichtlich darzustellen. Die Notation des obigen Beispiels wird etwa wie im Bild 6 gezeigt aussehen; sie orientiert sich an der in /2/ vorgestellten Programmiersprache OCCAM.
Die Dialogführung im Simulator wird verbessert. Beispielsweise wird es möglich, die zu einer Transition gehörige Bedingung in der SN700-Quelltextnotation (mathematische Schreibweise) anzeigen zu lassen.

Literatur

/1/ Programmieranleitung MRS 702/703. VEB Numerik „Karl Marx" Karl-Marx-Stadt
/2/ INMOS Limited: Occam Programming Manual. Prentice-Hall International, London 1984
/3/ IEC-Standard 65A (Secretariat) 49. Central Office of the IEC, Geneva, Schweiz, 1985

⊠ KONTAKT ☎

Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik, PSF 964, Karl-Marx-Stadt, 9091, Tel. 5 61 34 28, Dr. Kunke




```
SEQ GRADIENTENST
  STRT GRAD
  SEQ
    INCR_GRAD
    PASSIV
    COND NOT WARTEN AND TEMP<SOLL
    REPEAT
  END
END

SEQ VENTILSTEUERUNG
  VENT AUF
  COND NOT V RKUEHL OR ALARM
  VEN UMSCH
END

SEQ TEMPERATURREGELUNG
  HEIZ AKT
  COND NOT HEIZEN
  HEIZ PASS
  COND HEIZEN
  REPEAT
END

SEQ STEUERUNG
  UEBERWACH
  COND NOT HEIZEN EIN
  KONTROLLE
  ALT
    COND ALARM OR TEMP<MIN
    HUPEN
    OR COND NOT (ALARM OR TEMP<MIN)
  END
  HEIZ ENDE
END

SEQ AUSNAHME
  UEBERWACH
  COND ALARM
  HUPEN
END

SEQ PUMPENSTART
  VORBEREIT
  COND NOT ALARM AND NOT PUMPE EIN
  STRT_PUMP
  COND NOT PUMPE S
  ENDE_PUMP
  REPEAT
END

SEQ HEIZEN   (* HAUPTPROGRAMM *)
  H_EIN
  PAR
    SEQ
      VORBEREIT (* BEACHTE SEQUENZ PUMPENSTART *)
      COND ALARM OR NOT PUMPE
      PAR
        VENTILSTEUERUNG
        TEMPERATURREGELUNG
        GRADIENTENST
      END
    END
    STEUERUNG
  END
  VENTIL_ZU
END
```

Bild 4 Ablaufsteuerung für die Heizung einer Färbemaschine

Bild 6 Beschreibung des Steuerflusses mittels einer Sprache; Anlehnung der Strukturbeschreibung an Occam

Textverarbeitung auf Kleincomputern

Prof. Dr. Horst Völz, Berlin

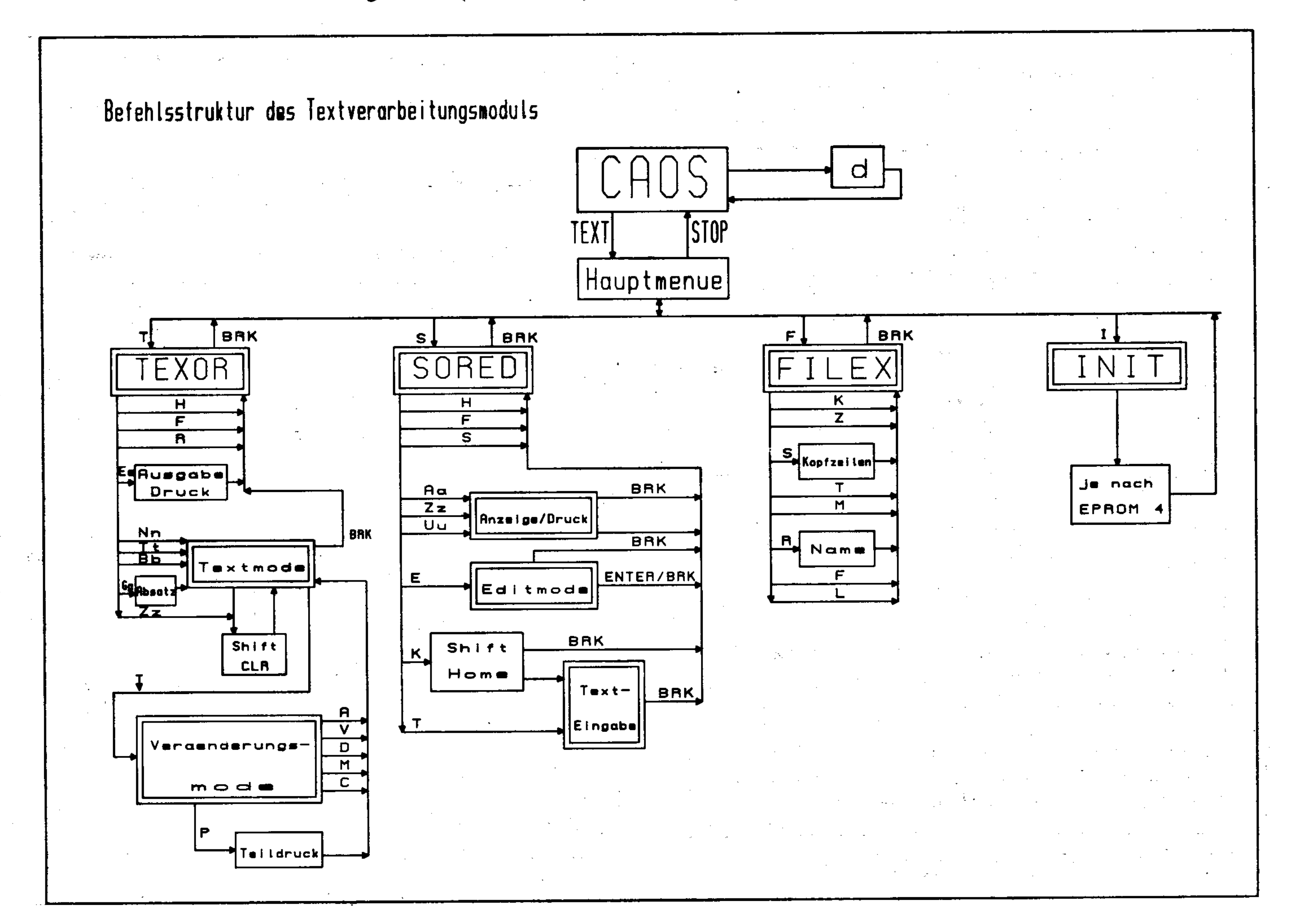
Entsprechend den Besonderheiten der KC 85/2 und KC 85/3 wurde eine neuartige Konzeption der Textverarbeitung entwickelt und realisiert. Sie zeichnet sich durch folgende Merkmale aus. Es existieren zwei miteinander koppelbare Teilsysteme – SORED und TEXOR. SORED arbeitet mit Quasizeilen und ist auf schnelles Sortieren und Suchen hin entwickelt. TEXOR ist ein vollständig bildschirmorientiertes Textverarbeitungssystem mit hohem Komfort. Die Kopplung beider und zusätzliche Hilfsfunktionen bietet ein drittes System FILEX.
Die Anzahl der Befehle wurde so gewählt, daß schnell – ohne viel zu lernen – mit den Systemen gearbeitet werden kann, und daß dann schrittweise die komplexe Handhabung bei der Arbeit mit dem System erworben werden kann. Der Befehlssatz ist auf Seite 119 zusammengefaßt.

SORED

Verwendet wird eine Zeilenstruktur. Damit die Zeilen mit Numerierung auf dem Bildschirm editiert werden können, ist die Zeilenlänge auf 32 Zeichen begrenzt. Um dennoch komplexe Zusammenhänge als Einheit sortieren und bearbeiten zu können, wurde ein Verbindungssonderzeichen ‾| eingeführt. Wird es am Ende einer Zeile eingegeben (bzw. bei 32 Zeichen automatisch erzeugt), so wird die folgende Zeile zur vorhergehenden gebunden. Auf diese Weise können u. a. Adressen, Literaturverzeichnisse usw. als Einheit für den Sortier- und Suchalgorithmus zusammengefaßt werden. Die zusammengehörenden Zeilen heißen Quasizeilen.
Das Sortieren ist allerdings nur vom ersten Zeichen einer Quasizeile beginnend möglich. Beim Sortieren werden große und kleine Buchstaben generell gleichwertig behandelt. Auch die Umlaute und ß werden richtig eingeordnet. Der Sortieralgorithmus ist durch Anwendung der Laufbefehle der CPU (U 880 – Z 80) extrem schnell. 32 K Text werden im worst-case-Fall in 7 Minuten sortiert!
Der Suchalgorithmus bei SORED besteht aus zwei entgegengesetzten Routinen. Es kann nach einer beliebig langen (oder kurzen) Zeichenkette gesucht werden. Die Zeichenkette kann dabei dont-care-Zeichen enthalten, so daß gleichermaßen z. B. Oxyd und Oxid gefunden werden können. Es werden dann alle Quasizeilen ausgegeben, die Ox■d enthalten (■ = dont-care). Die umgekehrte Zeichensuche gibt dagegen alle Quasizeilen aus, die nicht Ox■d enthalten. Da ein Sortierfile irgendwie gekennzeichnet und editiert werden muß, wurden auch hierfür Befehle vorgesehen. Eine frei wählbare Anzahl von Quasizeilen wird als „Kopf" bezeichnet und steht dann immer am Anfang mit allen sinnvollen Informationen – wie Name des File, Datum des letzten Editierens und erklärende Ergänzung zum File.
Das Editieren erfolgt in Quasizeilen voll bildschirmorientiert.
Der Druck und die Anzeige können in einem wählbaren Zeilenbereich, bei vorgebbarer Quasizeilenzahl je Seite (bzw. auf dem Bildschirm), mit und ohne Anzeige der Zeilennummer erfolgen. Die Eingabe dieser und aller anderen Parameter erfolgt durch Abfrage auf dem Bildschirm, d. h. im Nutzerdialog.

TEXOR

Infolge der relativ geringen scroll-Geschwindigkeit der KC 85/2 und KC 85/3



wurden ein kleines Bildfenster zur Eingabe (hohe Eingabegeschwindigkeit und schnelles Scrolling) und ein großes Bildschrimfenster für das Anzeigen und Editieren geschaffen. Der Bildschirm wird als Endloszeile verwendet. Nur Absätze (durch ENTER = CR) erzeugen eine echte, neue Zeile. Bei der Eingabe von Text wird immer der vom Bildschirm verschwindende Text in den Arbeitsspeicher übernommen. Der sichtbare Text (insbesondere bei der Anzeige) stellt also einen Textausschnitt dar. In einem File kann seitenweise vor und zurück und eine einzelne Zeile zurückgeblättert werden. Außerdem kann bei Eintritt sofort zu einem wählbaren Abschnitt bzw. an den Anfang oder das Ende des Textes gesprungen werden. Mit dem Text sind vielfältige Änderungen möglich, wie Absätze erzeugen bzw. aufheben, Textteile löschen, doppeln bzw. verlagern.
Auf dem Bildschirm selbst kann jederzeit beliebig Text überschrieben, Text eingefügt bzw. gestrichen werden. Als Sonderfunktionen sind Tabulator, Mittensatz und rechtsbündige Anordnung in der Zeile möglich.
Ein Text kann in einer beliebigen Zeilenlänge zwischen 10 und 225 Zeichen je Zeile ausgegeben werden. Die Formatierung erfolgt durch einen Randtest. Dabei wird automatisch eine Trennung durchgeführt. Wo eine Worttrennung nötig wird, bietet ein halbautomatischer Silbentrennalgorithmus einen Vorschlag an, der bestätigt werden kann oder bei dem das weiche Trennzeichen in einem gewissen Bereich nach rechts oder links verschoben werden kann. Die Druckerausgabe kann nach dem Randtest erfolgen. Dabei ist nach Wunsch Flatter- oder Blocksatz möglich.
Auch TEXOR enthält einen Suchalgorithmus für Zeichenketten.

FILEX

FILEX gestattet, einen SORED- in einen TEXOR-File bzw. umgekehrt zu wandeln. Mit ihm können in einfacher Weise Files auf Kassette gerettet bzw. von Kassette geladen werden. Auch ein Verbinden von Files ist möglich. Darüber hinaus enthält FILEX Möglichkeiten, einzelne Zeichen bzw. Zeichenketten im gesamten File zu modifizieren. Infolge des Wandels zwischen SORED und TEXOR lassen sich z. B. auch Zeilen, die nicht Quasizeilen sind und mehr als 32 Zeichen enthalten, sortieren. Es lassen sich aber auch Quasizeilen zusätzlich erzeugen, indem z. B. in TEXOR das Verbindezeichen eingefügt wird. Es lassen sich Teile von Quasizeilen umstrukturieren usw. Bei der Anwendererprobung hat sich gezeigt, daß hier in

Die wichtigsten Parameter und Befehle von TEXOR

1. Hauptmenü führt zu:
T: TEXVE
S: SORED
F: FILEX
I: INIT
STOP: CAOS

2. TEXVE läßt zu:
Fileeintritt:
N: neues File
B: Ende eines File (bottom)
T: Beginn eines File (top)
G: Gehe zum Abschnitt X
Anzeigen:
H: Menü (help)
F: Fileparameter
Ausgaben:
R: Randtest, bei Silbentrennung ist weicher Bindestrich ← und → in Grenzen verschiebbar
e: Ausgabe mit Flattersatz (exmit)
Z: Zeichensuche
Zeichensuche fortsetzen mit (Shift CLR)
Zeichensuche abbrechen mit BRK
Verlassen:
BRK: zum Hauptmenü
Veränderungsmodus:
↧a: Erzeuge Absatz
↧v: Verbinde (Absatz)
◇Text ↧d: Lösche Text (delete)
◇Text ↧p: Ausgabe Text (P: Blocksatz) (print)
◇Text◇ ... ↧c: Kopiere Text nach ↧
◇Text◇ ... ↧m: Verlagere (move) Text nach ↧
Fenster klein:
a, t, b, g, z
Fenster groß:
A, T, B, G, Z
Sonderzeichen:
(nur das erste gilt in einem Absatz)
△ Symmetrie = Setze in Zeilenmitte
╲ Setze Text hinter dem Zeichen rechtsbündig
□ Tabuliere auf die Positionen 4, 8, 12, 16 usw.
Silbentrennung:
Automatische Trennzeichen sind neben **Space**
!"#$%')*+,−./:;<=>?
(hinter ihnen erfolgt die Trennung).
Halbautomatische Trennung als Angebot
· hinter y
· hinter Vokal + einem Konsonant.
Der Verschiebebereich ist bei der Zeilenlänge Z:
$Z(1-1/8)-2$ bis Z
Die Zeilenlänge ist formatierbar von 10 bis 255 Zeichen.
Textmodus:
Alle Zeichen sind Eingaben.
Betätigung der Cursor-Führungstasten bewegt den Cursor sinngemäß auf dem Bildschirm.
INS, DEL, CLR, HOME – wie im Betriebssystem
Shift ↓ Vorblättern 1 Seite
Schift ↑ Zurückblättern 1 Seite
↑ Cursor am oberen Bildschirmrand – eine Zeile zurückblättern
Sonderzeichen entfernen
W Vernichten (löschen) von: ◇↧
L Löschen von: ~◇↧
nach @ werden alle Zeichen um 20 H verringert und damit als Steuerzeichen für Drucker erzeugt. CR beendet diese Steuerzeile.

3. SORED
Eingaben:
K: Kopf
Beginn erst nach wirklich? und J, BRK beendet
T: Texteingabe, BRK beendet
¯|: Sonderzeichen zum Verbinden von Zeilen zu einer Quasizeile am Ende einer Zeile
Anzeigen:
A: Ausgabe von Text
H: Menü (help)
F: Fileparameter
Z: Zeichensuche
(Für Zeichenkette don't care = Shift Space und zur Korrektur nur CLR)
U: Umgekehrte Zeichensuche
Zeilennummer und Verbindezeichen erscheinen bei A, Z, U. Dies ist für Edit notwendig. Bei a, z, u ohne Zeilennummer und Verbindezeichen.

Funktionen:
S: Sortiert den Text (erst nach wirklich? J)
E: Übergang in Editmode
Mit Cursor kann auf dem Bildschirm geändert werden. Änderungen werden nur dann übernommen, wenn für jede Quasizeile ENTER betätigt wird.
D: delete (löschen) von Zeilen, (am besten von hinten beginnen)

4. FILEX
Filearbeit:
R: Retten eines File
L: Laden eines File
A: Anfügen eines gespeicherten File an das im Rechner vorhandene (merge)
F: Fileparameter
Ändern:
S: in SORED-File mit festzulegender Zeilenzahl
T: in TEXOR-File
Ersetzen im File:
Z: von Einzelzeichen
K: von Ketten aus einem oder mehreren Zeichen
BRK: zurück ins Hauptmenü

5. INIT
Installiert für die wichtigsten Drucker

6. d = Diagnosesystem
Eintritt mit:
d: XXXX: (Startadresse)
BRK: Anhalten
C: Fortsetzen
ENTER: Abbrechen

7. HEX-Code der Sonderzeichen

Zeichen	HEX-Wert	Taste
ä	80	F1
ö	81	F2
ü	82	F3
ß	83	F4
╲	84	F5
↧	85	F6
Ä	86	Shift F1
Ö	87	Shift F2
Ü	88	Shift F3
□	89	Shift F4
◇	8A	Shift F5
△	8B	Shift F6
~	8C	Shift F7

Achtung! Die Kodes 7B bis 7F ergeben:
ä ö ü ß. Sie sind nicht über Tasten erreichbar.

8. Sortieren der Zeichen
Hierbei gilt das Alphabet, kleine und große Buchstaben sind gleichwertig
a = A = ä = Ä
o = O = ö = Ö
u = U = ü = Ü
s = ß
Vor den Buchstaben stehen in der folgenden Reihenfolge:
Space !"# $% & ' ( ) * + , − . / dann:
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
Nach den Buchstaben:
■ | ¯| ^ _
Die über Tasten nicht erreichbaren Kodes 7B bis 7F fehlen hier!

9. File-Aufbau
Jeder File beginnt mit 0.
Jeder Absatz bzw. jede Quasizeile endet mit 0.
Das Quasizeilensonderzeichen ist 5 DH. Das Ende des Files ist durch 0FFH gekennzeichnet.
Hinter dem Fileende, also nach dem ersten 0FFH, stehen weitere 2 Byte, die zum File gehören:
SORED die Kopfquasizeilenzahl
TEXOR 0FFH 0FFH
Die Zeilennumerierung bei SORED und die Abschnittanwahl bei TEXOR erfolgen über das Zählen der Nullen.
Auf der o. g. Basis kann die Kopfzeilenzahl in SORED leicht geändert werden, und zwar manuell z. B. mit MODIFY oder mittels der TEXOR-SORED-Wandlung. Sie ist folglich auch „illegal" für ein SORED-File in diesem Sinne anwendbar.

der Kombination eine Vielzahl von Varianten effektiv zusätzlich genutzt werden kann.

Vorhandene Varianten

Von SORED und TEXOR existieren je eine Bandversion, die etwa nur 2 KByte des Arbeitsspeichers verbrauchen. Teile des Programms werden an verschiedene Stellen des Bildwiederholspeichers bzw. in den Bereich unter 170H bei Laden verlegt. Trotz dieses geringen Speicherbedarfs bleibt die Leistungsfähigkeit der beiden Teilsysteme (je für sich) voll erhalten.
Neben den Bandversionen gibt es eine Version auf 8-K-EPROM. Hier sind neben FIL-EX, SORED und TEXOR noch ein Diagnosesystem und eine universelle Steuerung der V.24-Schnittstelle enthalten.
Außerdem besitzt der Modul TEXOR 2.2 weitere Verbesserungen, die insbesondere eine komplexe Steuerzeichengenerierung für Drucker ermöglichen.

Gleitpunkt-Arithmetik-Modul für U 880

Andreas Bogatz
Karl-Marx-Universität Leipzig,
Sektion Physik

Bei nahezu allen Anwendungen von Mikroprozessoren ist es erforderlich, arithmetische Operationen auszuführen, um z. B. Meßdaten mit Eingabeparametern zu verknüpfen und im Ergebnis Ausgabedaten zu erzeugen bzw. bestimmte Reaktionen auszulösen. Bei dem heute noch vorherrschenden Typ eines Mikroprozessors mit 8 Bit Verarbeitungsbreite sind solche arithmetischen Operationen wie Multiplikation, Division und alle Gleitpunkt- bzw. Festpunktoperationen mit mehr als 16 Bit Verarbeitungsbreite nicht Bestandteil des Befehlssatzes. Zur Lösung dieser Aufgaben muß daher ein entsprechender Software-Modul oder aber bei zeitkritischen Anwendungen eine Zusatz-Hardware (Multiplikationsfeld, Koprozessor) herangezogen werden. Bei allen nicht zeitkritischen Anwendungen wird man sich für eine Software-Lösung entscheiden.
Die nächste zu treffende Entscheidung bezieht sich auf den zur Anwendung kommenden Datentyp, speziell auf die Art und Weise der rechnerinternen Darstellung. Beschränkt man sich auf die Betrachtung numerischer Daten, so können diese in Fest- oder Gleitpunktformat vorliegen. Weiterhin kann ein Vorzeichen in Form einer Vorzeichen-Betrag-Darstellung oder einer 2er-Komplement-Darstellung berücksichtigt werden.

Programmbeispiele für die Realisierung von Arithmetikprogrammen im Dual-Festpunkt bzw. Dual-Gleitpunktformat wurden z. B. in /1, 2/ beschrieben. In beiden Fällen wird auf eine Rundung der Rechenergebnisse verzichtet, was sich positiv auf die Rechengeschwindigkeit auswirkt. Bei der praktischen Anwendung eines zu /1/ ähnlichen Programmpaketes (Bestandteil des MC80-BASIC) stellt sich jedoch die fehlende Rundung als eine Fehlerquelle bei längeren Rechnungen heraus.
Dazu kommen bei der Anwendung von dualen Gleitkommadarstellungen als weitere Fehlerquelle die Eingabe- und Anzeige-Konvertierungen, d. h. die BCD-Dual- und Dual-BCD-Wandlung. Dies liegt daran, daß sich gebrochene Zahlen nicht immer eindeutig von einer Zahlenbasis auf eine andere übertragen lassen. Dadurch kann es vorkommen, daß von den theoretisch auswertbaren 7 Dezimalstellen (entsprechend 24 Bit Mantisse) beim MC80 nur drei oder vier Stellen signifikant sind. So liefert z. B. das MC80-BASIC folgendes bei der Berechnung des gebrochenen Teils einer Variablen:
B = A − INT (A) ermittelt für
A = 1.00400 den Wert
B = 3.99948E-3.

Tafel 1 Zahlendarstellung der BCD-Arithmetik

Exponent	Mantisse			Beispielzahl
00	10	00	00	1.00000E-64
80	10	88	77	−1.08877E-64
40	53	00	99	5.30099
C3	34	12	23	−3.41223E+3
7F	87	65	43	8.76543E+63
Register/Stackzeiger				
B	C	H	L	
(IY +3)	(IY +2)	(IY +1)	(IY)	

Die Schnelligkeit dualer Algorithmen muß also mit einem sich mehr oder weniger auffällig fortpflanzenden Fehler der Rechnung erkauft werden.
Bei genauerer Betrachtung bieten sich zwei Lösungen an:
– Rechnung mit doppelter Genauigkeit und Rundung der Anzeige auf einfache Genauigkeit oder
– Rechnung mit einem BCD-Rechenpaket mit Rundung, da dann die Konvertierungen als Fehlerquelle entfallen.

Im ersten Fall kann die in /1, 2/ verwendete Registerarithmetik nicht mehr genommen werden, wodurch der Rechenzeitbedarf wegen der notwendigen Speicherrechnung stark ansteigt. Weiterhin wird der nahezu doppelte Speicherplatzbedarf für Variablen notwendig.
Dieser zunächst beschrittene Weg wurde daher verworfen. In der Folge entstand daher ein Gleitpunkt-BCD-Arithmetikmodul. Duale Meßwerte, wie sie etwa von A/D-Wandlern geliefert werden, lassen sich problemlos mit Hilfe von Konvertierungsprogrammen in die Rechnung einbeziehen, da es sich dabei um ganzzahlige Werte handelt, die keine Konvertierungsfehler hervorrufen.

Praktische Realisierung

Der Gleitpunkt-Arithmetikmodul realisiert eine Einadreßmaschine. Der erste Operand befindet sich beim Aufruf des Programmes in einem Gleitpunktakkumulator, gebildet von den Registern BCHL. Hierin wird auch das Ergebnis der Operation abgelegt. Der zweite Operand befindet sich auf einem arithmetischen Stack, gezeigert durch das Register IY.
Die Zuordnung der Ziffernstellen zu den Registern bzw. Speicherstellen vermitteln Bild 1 und Tafel 1. Der dezimale Exponent liegt zwischen −64 (OOH) und 63 (7FH). Die dezimale Mantisse besteht aus 3 Byte entsprechend 6 Dezimalstellen. Dabei steht der Dezimalpunkt nach der ersten Dezimalstelle und die Zahl Null wird durch die Mantisse 0.00000 dargestellt. Der somit beschreibbare Zahlenbereich ist also 0,+/−1.00000E-64 ... +/−9.99999E63 mit einer Auflösung von 0.00001.

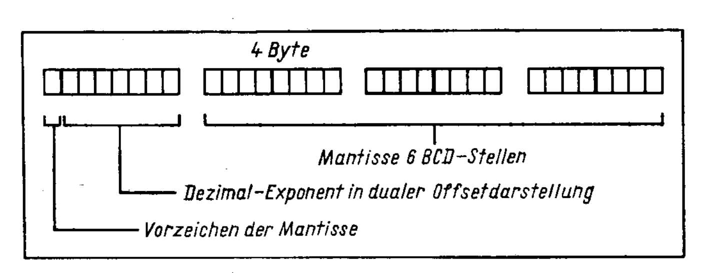


Bild 1 Interne Zahlendarstellung der BCD-Arithmetik

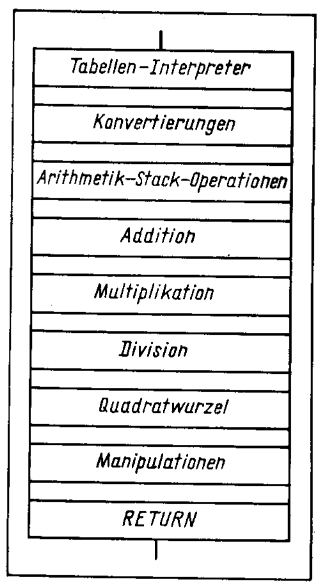


Bild 2 Struktur des Arithmetikmoduls

Der Aufbau eines arithmetischen Stack bietet gegenüber Rechenpaketen mit fester Speicheradressierung Vorteile:

- Unterbrechbarkeit und Wiedereintrittsfähigkeit und
- einfache Abarbeitung höherer Funktionen.

Der erste Punkt ist besonders in Zusammenhang mit Interruptroutinen wichtig, falls diese ebenfalls arithmetische Operationen benötigen. Konsequenzen einer Arithmetik mit festen Speicherstellen wurden in /3/ angegeben. Die dort aufgezeigten Auswege müssen jedoch als mangelhaft charakterisiert werden.
Der zweite Punkt erlangt Bedeutung beim Einsatz eines solchen Arithmetikmoduls zur Berechnung von Standardfunktionen und Anwenderfunktionen, z. B. innerhalb eines BASIC-Interpreters. Besonders unterstützt das Stackprinzip die Abarbeitung beliebiger mathematischer Formelausdrücke nach der Methode der umgekehrt polnischen Notation (RPN) /2/.
Ähnlich wie sich der zweite Operand auf einem arithmetischen Stack befindet, werden von den Programmteilen benötigte Hilfszellen kurzzeitig auf dem Prozessorstack angelegt, befinden sich also auch nicht auf festen Adressen.
Der Gleitpunkt-Arithmetikmodul enthält neben den Grundrechenarten Addition, Substraktion, Multiplikation und Division die Rechenoperationen Quadratwurzel, verschiedene Ladeoperationen, Arithmetik-Stack-Operationen und Fest-zu-Gleitpunkt sowie Gleit-zu-Festpunktkonvertierungen innerhalb der BCD-Darstellung. Weitere Konvertierungen erlauben die Überführung einer BCD-Ganzzahl in eine duale 20-Bit-Zweierkomplementdarstellung und umgekehrt. Weiterhin ist hervorzuheben, daß der Arithmetikmodul über eine ein-
daß der Arithmetikmodul über eine ein-
heitliche Einsprungstelle verfügt. Der Kode der abzuarbeitenden Operation wird dabei im Register A übermittelt. Tafel 2 vermittelt eine Übersicht zu den möglichen Operationen.
Bei einer Zahlenbereichsüber- oder -unterschreitung erfolgt eine Fehlermeldung durch ein gesetztes Carry-Flag. Das Ergebnis enthält in diesem Fall den größten bzw. kleinsten darstellbaren Wert, während im Akku eine Fehlermeldung übergeben wird:

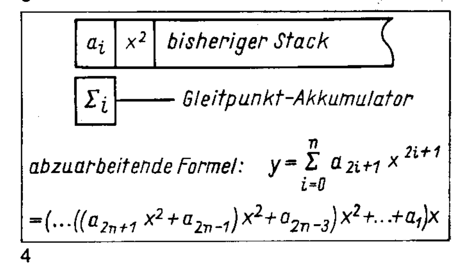
00H für Division durch Null
FFH für sonstigen Überlauf.

Diese Fehlermeldung kann durch das aufrufende Programm ausgewertet werden.

Tabellenkode-Interpreter

Der wesentliche Vorzug des Arithmetikmoduls besteht in der Möglichkeit, komplizierte arithmetische Ausdrücke auf einfache Weise zu berechnen. Dazu kann mit Hilfe eines Steuerkodes zwischen der Abarbeitung von Operation und der Abarbeitung von Operationskodetabellen umgeschaltet werden. Dazu betrachten wir folgendes Beispiel, das sowohl die Arbeit mit dem arithmetischen Stack als auch die Arbeit mit dem Tabellen-Interpreter veranschaulichen soll:

3



4

```
FN1:    LD   A,075H       ;STACK DEC, AKKU AUF STACK
        CALL ARI          ;AUSFüHREN
        LD   A,065H       ;MULTIPLIKATION
        CALL ARI          ;AUSFüHREN, ERGEBNIS X*X
        LD   A,06DH       ;AKKU AUF STACK
        CALL ARI          ;AUSFüHREN
        LD   BC,04050H    ;BCHL = 5.00000
        LD   HL,00000H
        LD   A,068H       ;ADDITION
        CALL ARI          ;AUSFüHREN, ERGEBNIS IN BCHL
        LD   A,079H       ;STACKZEIGER REGENERIEREN
        CALL ARI
        RET                            (Länge: 32 Byte)


FN2:    XOR  A            ;UMSCHALTCODE TABELLENARBEIT
        CALL ARI          ;AUSFüHREN
        DB   075H         ;STACK DEC, AKKU AUF STACK
        DB   065H         ;MULTIPLIKATION
        DB   06BH         ;STACK NACHLADEN
        DW   00000H       ;MIT KONSTANTE 5.00000
        DW   04050H
        DB   068H         ;ADDITION
        DB   079H         ;STACK INC
        DB   000H         ;TABELLENENDE --> RUEKSCHALTEN
        RET                            (Länge: 15 Byte)
```

Tafel 2 Operationskodes des Arithmetikmoduls

D0H:	Konvertierung BCD-Ganzzahl → Dual-Ganzzahl
B0H:	Konvertierung Dual-Ganzzahl → BCD-Ganzzahl
A6H...A0H:	Konvertierung Fest → Gleitpunkt, A6H – 6 Vorkommastellen
86H...80H:	Konvertierung Gleit → Festpunkt, 86H – 6 Vorkommastellen ... 80H – keine Vorkommastellen
7FH:	Stack increment, Stack lesen
7EH:	Stack increment, Akku auf Stack, Akku nachladen
7DH:	Stack increment, Akku auf Stack
7BH:	Stack increment, Stack nachladen
7AH:	Stack increment, Akku nachladen
79H:	Stack increment
77H:	Stack decrement, Stack lesen
76H:	Stack decrement, Akku auf Stack, Akku nachladen
75H:	Stack decrement, Akku auf Stack
73H:	Stack decrement, Stack nachladen
72H:	Stack decrement, Akku nachladen
71H:	Stack decrement
6FH:	Stack lesen
6EH:	Akku auf Stack, Akku nachladen
6DH:	Akku auf Stack
6BH:	Stack nachladen
6AH:	Akku nachladen
69H:	Stack mit Akku tauschen
68H:	Addition
67H:	Subtraktion
66H:	Differenz
65H:	Multiplikation
64H:	Division
63H:	Quadratwurzel
62H:	Vergleich
61H:	Ganzzahlanteil
60H:	Gebrochener Anteil
5FH:	Vorzeichenwechsel
5EH:	Betrag
5DH...01H:	Kodes eines eventuellen Funktionsmoduls
00H:	Umschaltkode Tabellenabarbeitung ein/aus

Bild 3 Polynomunterprogramm (ungerade); Belegung des arithmetischen Stack und Formel

Bild 4 Programmbeispiel für die Berechnung von $y = x^2 + 5$

Bild 5 Programmliste Arithmetikeintritt mit Tabellen-Interpreter

5

```
;----EINSPRUNGSTELLE ARITHMETIK----------------------------

ARI:    OR   A            ;UMSCHALTCODE TABELLENARBEIT ?
        JRNZ KON          ;NEIN: ZUM SPRUNGVERTEILER

;----TABELLENCODE-INTERPRETER------------------------------

TCI:    EX   DE,HL        ;ADRESSE DES ERSTEN CODES DER
        EX   (SP),HL      ;TABELLE HOLEN UND
        EX   DE,HL        ;UEBERGEORDNETES DE RETTEN
TC1:    LD   A,(DE)       ;CODE AUS TABELLE LESEN
        INC  DE           ;TABELLENZEIGER STELLEN
        OR   A            ;UMSCHALTCODE TABELLENENDE ?
        JRNZ TC2          ;NEIN: INTERPRETIEREN
        EX   DE,HL        ;RUECKKEHRADRESSE AUF STACK
        EX   (SP),HL      ;LEGEN UND UEBERGEORDNETES DE
        EX   DE,HL        ;REGENERIEREN
        RET               ;SPRUNG IN MASCHINENEBENE
TC2:    PUSH HL           ;RUECKKEHR IN TABELLEN-
        LD   HL,TC1       ;INTERPRETER RETTEN
        EX   (SP),HL

;----ERSTER VERTEILER--------------------------------------

KON:    CMP  ...

        USW.
```

```
5AH       : HILFSFUNKTION π/2 - X
59H       : HILFSFUNKTION 1/X
58H       : HILFSFUNKTION π

4FH...40H : UNGERADE POLYNOME MIT 1...16 SUMMANDEN,
            41H - 1 SUMMAND ... 4FH - 15 SUMMANDEN UND
            40H - 16 SUMMANDEN
3FH...30H : GERADE POLYNOME MIT 1...16 SUMMANDEN,
            31H - 1 SUMMAND ... 3FH - 15 SUMMANDEN UND
            30H - 16 SUMMANDEN
2FH...20H : ALLGEMEINE POLYNOME MIT 1...16 SUMMANDEN,
            21H - 1 SUMMAND ... 2FH - 15 SUMMANDEN UND
            20H - 16 SUMMANDEN

1FH       : ARCTAN(X)
1EH       : ARCSIN(X)
1DH       : ARCCOS(X)
1CH       : TAN(X)
1BH       : SIN(X)
1AH       : COS(X)
19H       : EXP(X)
18H       : 10^X
17H       : LN(X)
16H       : LOG(X)
```

Bild 6 Operationskodes des Funktionsmoduls

Es soll die Funktion

$Y = X \wedge 2 + 5$

ermittelt werden. Vorausgesetzt, der arithmetische Stack ist definiert und der Operand befindet sich im Gleitpunktakkumulator BCHL, dann können die in Bild 4 angegebenen Programme diese Funktion erfüllen. Hierbei ist ersichtlich, daß die Tabelleninterpretation wesentlich weniger Speicherplatz beansprucht.

```
;----ARCTAN-----------------------------------------

ATN:    CALL UVG        ;VERGLEICHE ZAHL MIT 1
        JRC  AT2        ;SPRUNG,FALLS GROESSER
AT1:    XOR  A          ;TABELLENARBEIT
        CALL ARI        ;ANWAEHLEN
        DB   048H       ;UNGERADES POLYNOM,8 SUMMANDEN
        DW   05406H     ;-4.05406E-3
        DW   0BD40H
        DW   08612H     ; 2.18612E-2
        DW   03E21H
        DW   09099H     ;-5.59099E-2
        DW   0BE55H
        DW   04200H     ; 9.64200E-2
        DW   03E96H
        DW   09085H     ;-1.39085E-1
        DW   0BF13H
        DW   09465H     ; 1.99465E-1
        DW   03F19H
        DW   03298H     ;-3.33298E-1
        DW   0BF33H
        DW   00000H     ; 1.00000
        DW   04010H
        DB   000H       ;ENDE TABELLENARBEIT
        RET
AT2:    XOR  A          ;RESET CARRY
        BIT  7,B        ;TEST VORZEICHEN
        PUSH AF         ;MERKEN
        CALL ARI        ;TABELLENARBEIT ANWAEHLEN
        DB   05EH       ;BETRAG
        DB   059H       ;1/X
        DB   01FH       ;ARCTAN
        DB   05AH       ;π/2 - X
        DB   000H       ;ENDE TABELLENARBEIT
        POP  AF         ;GEMERKTES VORZEICHEN
        RZ              ;FERTIG,FALLS POSITIV
        LD   A,080H     ;SONST: VORZEICHEN UMKEHREN
        XOR  B
        LD   B,A
        RET
```

Bild 7 Programmliste für die Funktion arctan(x)

Um die Funktionsweise des Tabellen-Interpreters zu veranschaulichen und eventuell anderen Anwendungsbeispielen zu eröffnen, soll dieser Programmteil hier angegeben werden.
Bild 2 zeigt die Struktur des Gesamtmoduls und Bild 5 zeigt den Programmanfang mit dem Tabellen-Interpreter. Das Gesamtprogramm besteht aus den einzelnen Operationsmoduln, zwischen denen sich regelmäßig angeordnete Sprungverteiler befinden, die den im Akku enthaltenen Operationskode analysieren. Der letzte Sprungverteiler verzweigt, falls kein gültiger Operationskode ermittelt wurde, auf den letzten Speicherplatz des Programmpaketes. Dort steht ein RETURN-Befehl (C9H), wodurch bei ungültigem Operationskode ins aufrufende Programm zurückgekehrt wird. Dadurch erhält der Anwender des Programmpaketes die Möglichkeit, das Programm durch weitere Operationen zu ergänzen. Dazu ist es lediglich notwendig, ab einschließlich dieses RETURN-Befehls einen weiteren Sprungverteiler anzufügen und die entsprechenden, dazugehörigen Programmteile zu notieren.

Erweiterungen

Ein solches erweiterndes Programmpaket wäre z. B. ein Paket der Standardfunktionen exp(x), $10 \wedge x$, ln(x), log(x), sin(x), cos(x), tan(x) und arctan(x). Allen diesen Funktionen ist gemeinsam, daß sie mit Hilfe von Polynomen ermittelt werden können. Ebenso führen empirisch ermittelte Zusammenhänge, die z. B. eine Eichkurve beschreiben, häufig auch auf Polynome.
Es liegt daher nahe, die Abarbeitung von Polynomen zur Basis eines Gleitpunktfunktionsmoduls zu erheben. Dieser Weg führt zu drei Typen von Polynom-Unterprogrammen zur Unterstützung der Abarbeitung höherer Funktionen:

gerades Polynom
ungerades Polynom
allgemeines Polynom.

Bild 6 zeigt eine Übersicht zu den Operationskodes des Gleitpunktfunktionsmoduls. Für die Polynomunterprogramme wird die Anzahl der Koeffizienten ebenso wie die Art des Polynoms im Operationskode verschlüsselt.
Als Beispiel zeigt Bild 3 die Belegung des arithmetischen Stacks während der Abarbeitung eines ungeraden Polynoms sowie die dabei abzuarbeitende Formel.
Abschließend soll anhand der Funktion arctan(x) die Nutzung der Polynomunterprogramme veranschaulicht werden. Bild 7 zeigt das dafür notwendige Programm, wobei die Koeffizienten aus /2/ übernommen wurden und die Funktion nach den Formeln

$y = \mathrm{sign}(x) * \arctan /x/$
für $/x/ <= 1$
$y = \mathrm{sign}(x) * (\pi/2 - \arctan 1//x/)$
für $/x/ \geqq 1$

berechnet wird.

Literatur

/1/ Philippow, I.; Fritz, K.-D.; Roth, M.: Basisalgorithmen für Prozeßinformationsverarbeitung mit U880. Radio, Ferns., Elektron., Berlin 31 (1982) 5, S. 292–294

/2/ Lampe, B.; Jorke, G.; Wengel, N.: Algorithmen der Mikrorechentechnik. 2. Auflage VEB Verlag Technik Berlin 1983

/3/ Hennig, D.: Softwarefehler bei Interruptbetrieb. Radio, Ferns., Elektron., Berlin 34 (1985) 11, S. 694

Emulator für Einchipmikrorechner U 88xx

Für die Einchipmikrorechnerfamilie M 88xx vom VEB Mikroelektronik „Karl Marx" Erfurt wurde ein Einkartenemulator entwickelt, der folgende technische Daten hat:

Leiterkartenformat
etwa 200 mm × 240 mm
Kopplung zum Monitorrechner
seriell, IFSS, galvanisch getrennt, Übertragungsrate 9,6 kBaud
Monitorrechner
BC A 5120 (UDOS), MC 80.20, MC 80.30
Stromversorgung
5 V, etwa 1,5 A
Kopplung zum Anwender
über 40poligen CPU-Schaltkreis-Adapter
Anwenderspeicher
4 KByte
Haltepunktspeicher
4 Kbit

Die Monitorsoftware für den BC A 5120, MC 80.20 und MC 80.30 ist über Diskette bzw. Magnetband ladbar.

Kommandoübersicht des Monitors:

B ADR Break setzen, löschen
D ADR Anzeige/Änderung Programmspeicher

G	ADR	Start der Programmausführung
GE		EMR-Programm Monitorrechner-Emulator
I	EID	Interrupt zulassen/sperren
M		Monitor-Break
N		Ausführung eines Befehls
P		Anzeige/Änderung PC
R		alle Register anzeigen
R	ADR	lesen/ändern Registerinhalt
S		Statusanzeige: PC, Register
SA		EMR-Programm Emulator-Monitorrechner
XD	ADR	ext. RAM lesen/schreiben

Die Dokumentation für die Hardware und die Bedienungsanleitung sind Bestandteile der Nachnutzung. Unbestückte, durchkontaktierte Leiterkarten stehen in begrenzter Anzahl zur Verfügung.

⊠ KONTAKT ☏

Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik, Wissenschaftsbereich Computertechnik, Am Ehrenberg, Ilmenau, 6300. *Prof. Dr. M. Roth*

Zu einem Interruptproblem beim U880

Roland Hahn, Dr. Hans-Joachim Gasse
Institut für Energetik

Der leistungsfähige Interruptmode 2 des Z80 hat viel zum weltweiten Erfolg des Systems beigetragen. Hier soll über einen wenig bekannten (oder besser: fast vergessenen) Fehler berichtet werden, der stochastische, selten auftretende und sehr schwer erklärbare Fehlfunktionen des Rechners zur Folge haben kann. Beispielsweise beim Zugriff auf globale Variable muß der Interrupt gesperrt werden. Ist dabei nicht bekannt, ob der Interrupt erlaubt oder verboten ist, so soll entsprechend /1/ der Zustand des Interruptflipflops mit dem Befehl **LD A, I** abgefragt und anschließend wiederhergestellt werden. Dieser Befehl (aber auch der Befehl **LD A, R)** kopiert den Zustand des IFF2 in das Parityflag, das mit einem bedingten Sprung getestet werden kann. Tatsächlich mußte festgestellt werden, daß dieses Vorgehen bisweilen einen gesperrten Interrupt vortäuschte. Eine genauere Untersuchung zeigte, daß bei der Anmeldung eines Interrupts während dieses Befehls das Parityflag falsch gesetzt wird. Die genaue Erklärung kann sicher nur der Entwickler des Schaltkreises geben, wahrscheinlich aber geschieht folgendes: Zu Beginn des letzten Taktes des Befehls **LD A,I** (**LD A,R**) wird die Interruptteilung abgefragt und intern IFF1 und IFF2 zurückgesetzt, die Übertragung in das Parityflag führt der Befehl erst danach aus. Damit ist die Interrupterlaubnis (denn sonst hätte ja der Interrupt nicht auftreten können) falsch gerettet worden. Zunächst wurde ein Exemplarfehler vermutet, dann ein Fehler bei der Übernahme von einem Hersteller zum anderen. Erst zuletzt wurde eine Bemerkung in einer sehr frühen Zilog-Dokumentation /2/ entdeckt, die diese Eigenschaft des **LD A,I**-Befehls erwähnt. Damit ist die Abfrage des Interruptzustandes mit diesem Befehl und überhaupt unmöglich, denn das Auftreten eines Interrupts genau auf diesem Befehl ist nicht auszuschließen.

Zum softwaremäßigen Überwinden dieses Problems ist es notwendig, an allen Stellen eines U880-Programmes zu wissen, ob der Interrupt erlaubt ist oder nicht. Insbesondere bei großen, von mehreren Bearbeitern zu erstellenden Programmsystemen ist bei der Definition der Interfaces global benutzter Teil- bzw. Unterprogramme, die per Interruptsperre zu schützende kritische Regionen enthalten, zu empfehlen:

– für jeden Eintrittspunkt muß eindeutig festgelegt sein, ob er mit erlaubtem oder unerlaubtem Interrupt aufzurufen ist, oder

– Unterprogramme, die sinnvoll sowohl bei erlaubtem als auch bei unerlaubtem Interrupt abgearbeitet werden können (z. B. Aufruf aus Rahmenprogramm und aus Interruptserviceroutine), müssen den aktuellen Zustand des IFF2 als Parameter mitgeteilt bekommen.

Es wurde auch eine hardwaremäßige Lösung gefunden (Bild 1). Dabei wird während des **LD A,I**-Befehls der Interrupt verboten. Dazu wird am höchstpriorisierten Peripherieschaltkreis das IEI-Signal während des letzten Befehlszyklus ausgeschaltet. Die Dekodierung der beiden Operationscodes ED und 57 (bzw. 5F) an der Rückflanke von M1 genügt allerdings nicht. Es sind auch die Befehlsfolgen

```
SET 5,L   CB ED
LD  D,A   57
und
SET 1,E   CB CB
LD  A,I   ED 57
```

(wenn auch etwas akademisch) denkbar. Im ersteren Fall soll IEI nicht ausgeschaltet werden, was durch Dekodieren des CB erreicht werden kann, im zweiten Fall muß es aber erfolgen.

Die Schaltung erzeugt beim Operationscode ED auch das Signal IEP, mit dem die IEI-Signale aller Peripherieschaltkreise eingeschaltet werden. Damit wird das bekannte RETI-Problem /3/ überwunden. Die Dekodierung des Operationscodes kann besonders einfach mit einem elektrisch programmierbaren Festwertspeicher (256 × 4 Bit, Typ MH74S287 von Tesla) vorgenommen werden.

Bild 1 Hardware-Lösung

Literatur

/1/ E. L.: Softwarefehler bei Interruptbetrieb. Radio, Ferns., Elektron. 35 (1986) 6, S. 342

/2/ Z80-Assembly Programming Manual. ZILOG 1977, S. 48

/3/ Prikryl, J.: Interrupt-Behandlung im Z80-System und Konsequenzen für die E/A-Erweiterung. Elektronik 13/80. S. 55

V.24-Modul M003

Serielle Schnittstelle für KC 85/2 und KC 85/3

Klaus-Dieter Kirves
VEB Mikroelektronik „Wilhelm Pieck"
Mühlhausen

Eine wesentliche Erweiterung der Einsatzmöglichkeiten der Kleincomputer ergibt sich beim Anschluß peripherer Geräte. Das wichtigste Gerät, welches für viele Anwendungen unerläßlich ist, ist ein Drucker. Er ermöglicht unter anderem den Listenausdruck selbsterstellter Programme oder die Protokollierung von Ergebnissen. Textverarbeitungssysteme kommen ohne einen Drucker oder eine Schreibmaschine nicht aus. Voraussetzung zum Anschluß ist ein standardisiertes Interface. Sehr weit verbreitet und bequem ist ein serielles Interface nach der TGL 29077/01/02 (V.24). Über vier Signalleitungen wird ein Datenaustausch bis zu 15 Metern ermöglicht. Die Datenübertragung erfolgt asynchron mit 5 bis 8 Datenbits, einem Startbit und 1 bis 2 Stopbits. Ein Paritätsbit kann zur Datenprüfung mit übertragen werden (Bild 1). Für die Kleincomputer KC 85/2 und KC 85/3 des VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen wird die V.24-Schnittstelle in Form eines Moduls angeboten. Dieser Modul M003 V.24 kann in einem der beiden Modulschächte des Grundgerätes oder in einem Modulschacht eines Aufsatzes kontaktiert werden. Der Modul enthält zwei unabhängige V.24-Kanäle. Jeder Kanal verfügt über zwei fünfpolige Diodenbuchsen, denen je eine Datensende- und Datenempfangsleitung mit Quittungsleitungen zugeordnet sind. Tafel 1 zeigt die Belegung der Diodenbuchsen.

Tafel 1 Anschlußbelegung der Diodenbuchsen

Anschlußnummer der Diodenbuchse	Leitungsnummer	Signalbezeichnung	Signalrichtung
2	102	Signal- und Betriebserde	
3	103	Sendedaten TXD	Ausgang
1	104	Empfangsdaten RXD	Eingang
4	106	Bereit zum Senden CTS	Eingang
5	108	Betriebsbereitschaft DTR	Ausgang

Aufbau

Der DART-Schaltkreis U 8563 /1/ bildet den Kern des V.24-Moduls. Die Taktversorgung des DART erfolgt programmierbar mittels eines CTC-Schaltkreises U 857 D aus dem Systemtakt von etwa 1,75 MHz. Sende- und Empfangstakt eines Kanals sind gleich. Die für die Aus- und Eingangstreiberstufen erforderliche negative Versorgungsspannung wird mittels eines eisenlosen Transverters mit einem Timerschaltkreis B 555 D im Modul erzeugt. Der V.24-Modul beinhaltet weiterhin eine Modulsteuerung entsprechend dem Modulkonzept der Kleincomputer KC 85/2 und KC 85/3. Somit ist der Modul online und offline schaltbar. Mehrere gleichartige V.24-Module auch mit gleichen Ein- und Ausgabeadressen von DART und CTC können quasi gleichzeitig im System betrieben werden. Ein Kleincomputer ohne Aufsatz könnte somit über zwei V.24-Module mit vier weiteren Computern oder anderen Geräten gekoppelt werden. Bild 2 zeigt das Blockschaltbild des V.24-Moduls.

Software

Allgemeines zur Software

Der V.24-Modul kann nur in Verbindung mit entsprechenden Programmen, z. B. Druckertreiberroutinen, genutzt werden. Da für jeden Einsatzfall, speziell für verschiedene Drucker, unter-

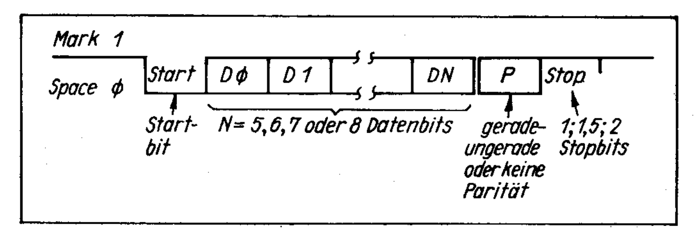


Bild 1 Asynchrones Datenformat

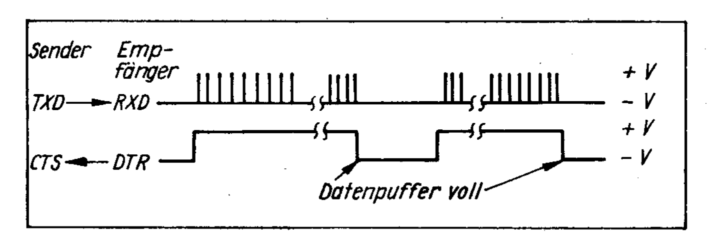


Bild 3 Zeitdiagramm des Hardware-Protokolls

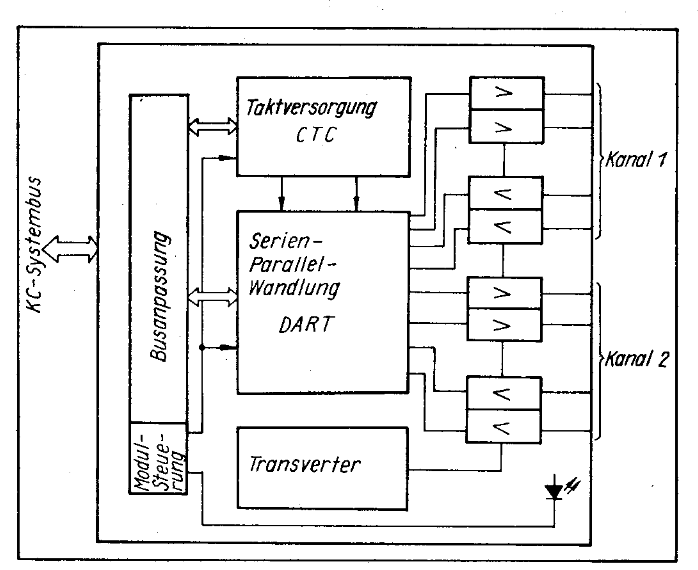


Bild 2 Blockschaltbild des V.24-Moduls M003

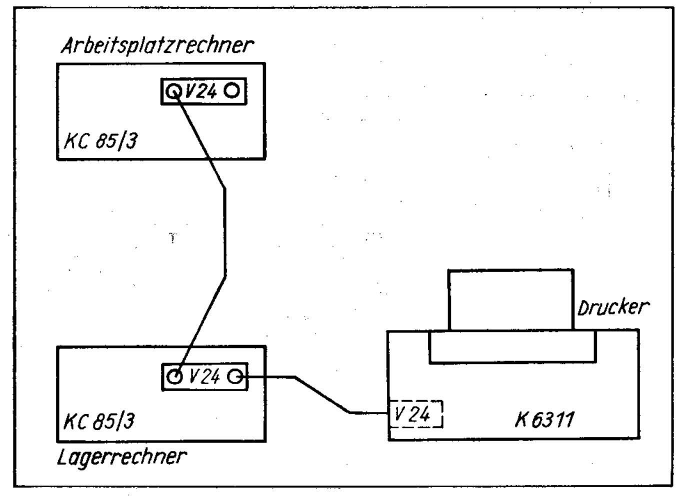


Bild 4 Kopplung zweier KC 85/3 über V.24-Module

schiedliche, dem System angepaßte Programme benötigt werden, wird neben dem V.24-Modul eine Kassette mit mehreren V.24-Treiberprogrammen angeboten. Alle auf der Kassette befindlichen V.24-Treiberroutinen arbeiten im Hardware-Protokoll (Bild 3). Die Übertragung wird über eine zusätzliche Quittungsleitung in jeder Richtung gesteuert. Der Datensender eines Kanals kann über den zugehörigen CTS-Eingang vom DTR-Ausgang des Empfängers gesperrt werden, wenn der Empfangspuffer gesperrt bzw. keine Empfangsbereitschaft vorhanden ist.

Drucker- und Schreibmaschinenansteuerungsprogramme

Die Kassette C0171 enthält Programme für die Drucker K 6303, K 6311, K 6312 sowie für die Schreibmaschinen S 6005 und 6010 bzw. Schreibmaschinen mit gleicher Ansteuerung. Die Treiberroutinen werden in den Kleincomputer in den Speicherbereich BAOOH bis BBFFH /2/ geladen und belasten somit den verfügbaren Speicherbereich beim BASIC nicht. Nach dem Laden (auch von BASIC aus mit der BLOAD-Anweisung) starten sie automatisch, schalten den V.24-Modul online, initialisieren den V.24-Kanal und ggf. den Drucker. Dabei sind die Parameter Modulschacht, Modulkanal, Übertragungsrate und Datenformat fest eingestellt. Die Parameter können aber vom Anwender entsprechend anderen Gegebenheiten leicht verändert werden. Die neu generierte Version kann wieder auf Kassette abgespeichert werden und ist damit ebenso handhabbar wie das Original. Aufgerufen werden können die Drucker vom BASIC aus über LIST # 2 oder PRINT # 2 (wahlweise auch # 3). Ebenso ist es möglich, über Tastenkombination den Drucker parallel zum Bildschirm zu schalten, das heißt, alle Ausgaben, die bisher ausschließlich auf dem Bildschirm dargestellt wurden, werden parallel dazu ausgedruckt. Weiterhin werden für alle Gerätetreiberroutinen noch erweiterte COPY-Funktionen, welche über Tastenkombination oder Programm aufrufbar sind, angeboten. Für Schreibmaschinen gibt es eine SCREEN-COPY-Funktion, bei deren Aufruf alle druckbaren (ASCII-) Zeichen aus dem Bildwiederholspeicher ausgeschrieben werden. Bei Matrixdruckern ermöglicht eine HARD-COPY-Funktion die punktweise Wiedergabe des Bildschirminhaltes, was die Grafikfähigkeit der Kleincomputer unterstützt.
Es gelten folgende Besonderheiten:

- Thermodrucker K 6303
- HARD-COPY quer zur Papiertransportrichtung mit 240 × 320 Punkten
- Matrixdrucker K 6311
  - Einstellung auf Schriftart 100 Zeichen/Zeile
  - linker Heftrand
  - Format 68 Zeilen auf A4-Seite
  - HARDCOPY mit 256 × 320 Punkten
- Matrixdrucker K 6312
  - linker Heftrand
  - Format 72 Zeilen je Seite mit Transportlochung
  - HARDCOPY mit 512 × 640 Punkten (1 Bildschirmpunkt = 4 gedruckte Punkte)
- Schreibmaschinen S 6005/S 6010
  - Transformation der Umlaute entsprechend Standardzeichensatz
  - SCREEN-COPY 40 Zeichen in 32 Zeilen

Kleincomputer mit V.24-Eingang

Neben der Anwendung des V.24-Moduls als Ausgabeschnittstelle ist auch die Eingabe von Daten über den Modul möglich. Als Datensender sind zum Beispiel intelligente Tastaturen, Meßgeräte oder Schreibmaschinen, denkbar. Die Software-Kassette CO171 enthält eine V.24-Eingaberoutine in zwei Adreßversionen: für den Bereich BAOOH bis BAFFH, wie die Druckertreiber und als Ergänzung zu den Druckertreibern, im Bereich BCOOH bis BCFFH. Die Eingabe kann beim BASIC mit LOAD#2 (#3) bzw. INPUT#2 (#3) erfolgen. Es ist aber auch möglich, sämtliche Systemeingaben über den V.24-Modul (externe Tastatur) zu realisieren. Eine Übertragung mit genormter Übertragungsrate ist hierbei jedoch auf Grund der Taktfrequenz des KC 85/2 bzw. KC 85/3 nur bis 2400 Bit/s möglich.

V.24-Duplex-Schnittstelle

Eine weitere wichtige Anwendungsvariante des V.24-Moduls ist eine Duplexschnittstelle zu anderen Rechnern. Mit ihr erschließen sich völlig neue Anwendungsbereiche. Als Beispiele seien hier genannt:

- Kopplung zweier Kleincomputer zum Daten- bzw. Programmaustausch
- relativ bewegliches Datenerfassen und -vorverdichten mittels Kleincomputer und Datentransfer zum größeren Rechner zur Weiterverarbeitung
- Zugriff vom Kleincomputer auf die Peripheriegeräte anderer Rechner (z. B. Drucker, Plotter, Digitalisiergerät, Massenspeicher)
- Einsatz des Kleincomputers als Grafikdisplay, z. B. am Bürocomputer.

Die Kassette enthält ebenfalls zwei Versionen des Duplextreibers. Zur Übertragungsrate gilt das oben gesagte, mit der Ergänzung, daß bei der Kopplung zweier Kleincomputer KC 85/2 bzw. KC 85/3 die maximale Übertragungsrate etwa 5400 Bit/s (nicht genormt!) betragen kann.
Auf der Leipziger Frühjahrsmesse 1986 wurde ein Koppelbeispiel zweier KC 85/3 über V.24-Module gezeigt (Bild 4). Auf einem KC 85/3 (Arbeitsplatzrechner) lief ein Statikprogramm. Der Konstrukteur konnte nach erfolgter Berechnung über die V.24-Kopplung im Lagerrechner „nachfragen", ob die benötigten Teile am Lager sind. Weiterhin konnte er die berechnete Menge vom Lagersimulationsprogramm abziehen. Der Lagerrechner druckte daraufhin einen Materialentnahmeschein. Am Lagerrechner sind auch Materialein- und -ausgaben möglich. Beide Programme waren bis auf die Duplexschnittstelle in BASIC geschrieben. Im BASIC-Lagersimulationsprogramm erfolgt auch die Zugriffssteuerung zwischen eigener Tastatureingabe und Kopplung zum Konstruktionsrechner.

Literatur
/1/ Kieser, H.; Meder, M.: Mikroprozessortechnik – Aufbau und Anwendung des Mikroprozessorsystems U 880, VEB Verlag Technik, Berlin 1985
/2/ Domschke, W.: Der Kleincomputer KC 85/3, Mikroprozessortechnik 1 (1987) 2

Datenaustausch KC 85/2 – MRES A 5601

Um auf dem MRES A 5601 erstellte Maschinen- oder Quellprogramme auf dem KC 85/2 nutzen zu können, wurde eine Übertragungsmöglichkeit über zwei V.24-Module des KC-Systems entwickelt. Ein Adapter ermöglicht es, die KC-Module im K-1520-System zu betreiben. Ein Programm von etwa 2 KByte Länge für das MRES gestattet den byteweisen oder blockweisen Datenaustausch. Das Programm erlaubt weiterhin den Austausch von Quellprogrammen zwischen dem RAM und den peripheren Geräten des MRES. Dieser Teil kann auch unabhängig von der V.24-Treiberroutine genutzt werden und stellt eine wesentliche Erweiterung der Dienstprogramme des A 5601 dar.
Angeboten werden Unterlagen zum Moduladapter, Quellprogramm zur V.24-Kopplung, MRES-Systemprogramm zum Datenaustausch zwischen V.24-Kanal und RAM bzw. peripheren Geräten und RAM (übersetzt je nach Speicherkonfiguration) sowie KC-85-Treiberprogramme für Datenaustausch über V.24-Modul.

Schröter

✉ KONTAKT ☏
VEB Mikroelektronik „Wilhelm Pieck"
Mühlhausen, Eisenacher Straße 40, 5700;
Tel. 587576

Anschluß von Druckern und elektronischen Schreibmaschinen an robotron-Kleincomputer

Im Sortiment der Erweiterungsbaugruppen für die robotron-Kleincomputer KC 85/1 und KC 87 werden Anschlußmodule für Drucker und elektronische Schreibmaschinen angeboten, die einen V.24-Interfaceanschluß besitzen. Diese Module sind hardwareseitig identisch und unterscheiden sich nur durch den Steckverbinder am Anschlußkabel und das auf dem EPROM des Moduls eingelesene Druckertreiberprogramm. Die Tafel gibt eine Übersicht der Zuordnung von Modultypen zu den geeigneten Druckern bzw. Schreibmaschinen. Ab 3. Quartal 1987 ist für den Druckermodul 690025.2 und den Schreibmaschinenmodul 690021.1 ein verbessertes und einheitliches Druckertreiberprogramm im Angebot. Damit sind alle Drucker und elektronischen Schreibmaschinen mit V.24-Schnittstelle, bei Anpassen des Steckverbinders, unter folgenden Bedingungen anschließbar: DTR-Protokoll (Hardware-Protokoll); 9600 Baud; 8 Datenbit; kein Paritätsbit, 1 Stopbit.
Als Option können noch folgende Parameter durch ein kundeneigenes Druckertreiberprogramm, basierend auf einem universellen Programm auf der Programmkassette R 0171 des VEB Robotron-Vertrieb, realisiert werden: 1 Startbit; 7 oder 8 Datenbit; gerades, ungerades oder kein Paritätsbit; 1 oder 1½ oder 2 Stopbit; 50 bis 9600 Baud.

Dr. G. Kleinmichel

Übersicht der Modultypen

Modul	Technische Hauptparameter	anschließbare Drucker bzw. Schreibmaschinen
Druckermodul 690 006.8	Übertragungsrate: 1200 Baud, 1 Startbit, 7 Datenbit, 1 Paritätsbit (gerade) 1 Stopbit; Diodenstecker	Thermodrucker
Druckermodul 690 025.2	Übertragungsrate: 9600 Baud 1 Startbit, 8 Datenbit, kein Paritätsbit, 1 Stopbit, Buchsenleiste 222-26, TGL 29331/04	Nadeldrucker: K6311①, 12①, 13③, 14③, 16①, Thermodrucker K6304③, Seriendrucker 1152, 1157 Schreibmaschine: S6005, 10③, 6006③ 6010②, 6120③, 6130③
Schreibmaschinenmodul 690 021.1	Übertragungsrate: 9600 Baud, 1 Startbit, 8 Datenbit, kein Paritätsbit, 1 Stopbit, Steckerleiste Cannon DB-255	Nadeldrucker K6311①, 12①, 13, 14, 16①, Thermodrucker K 6304 Seriendrucker SD1152③, SD1157③ Schreibmaschine: S6005.10①, 6006③, 6010③, 6120②, 6130

① mit entsprechendem Steckverbinder
② mit Änderung der Steckverbinderbelegung
③ mit Austausch des Steckverbinders

JUNOST 401B als Monitor für KC 85/3

Beim Einsatz der Kleincomputer KC 85/3 bzw. KC 85/2 wird aus ökonomischen und auch aus Platzgründen häufig auf Schwarz-weiß-Portable als Ausgabegerät zurückgegriffen. Da der Kleincomputer neben dem HF-Ausgang auch über einen kombinierten FBAS-RGB-Anschluß verfügt, bietet sich für Fernsehgeräte die Nachrüstung mit einem BAS-Eingang an. Neben einer erheblichen Verbesserung der Bildqualität, welche z. B. die Voraussetzung für eine 80stellige Textverarbeitung /1/ ist, kann zusätzlich eine lautstärkegesteuerte Tonausgabe über den Lautsprecher des Fernsehgerätes erfolgen.
In /2/ wird ein einfacher BAS-Anschluß für den Junost 402B mit manueller Umschaltung angegeben. Bild 1 zeigt eine Variante mit automatischer Umschaltung des Bild- und des Tonkanals im FS-Gerät. Zur Umschaltung der Kanäle dient ein 4fach-CMOS-Analogschalter V4066. Zwei Schalter werden jeweils als ein Umschalter verwendet. Die Umschaltspannung des KC beträgt 1 V an 75 Ohm. Mit T1 erfolgt die Pegelanpassung an den CMOS-IS und mit T2 die Negation für die Umschalter. Der Widerstand R5 stellt einen zulässigen Gleichspannungspegel für die Analogschalter sicher.
Der Aufbau kann auf einer kleinen Universalleiterplatte erfolgen. Für die entprechenden Leiterzüge im FS-Gerät werden abgeschirmte Kabel verwendet. Im aufgebauten Muster wurde der Umschalter zwischen Teleskopantenne und Antennenbuchse entfernt und dafür eine 5polige Diodenbuchse eingepaßt. Die

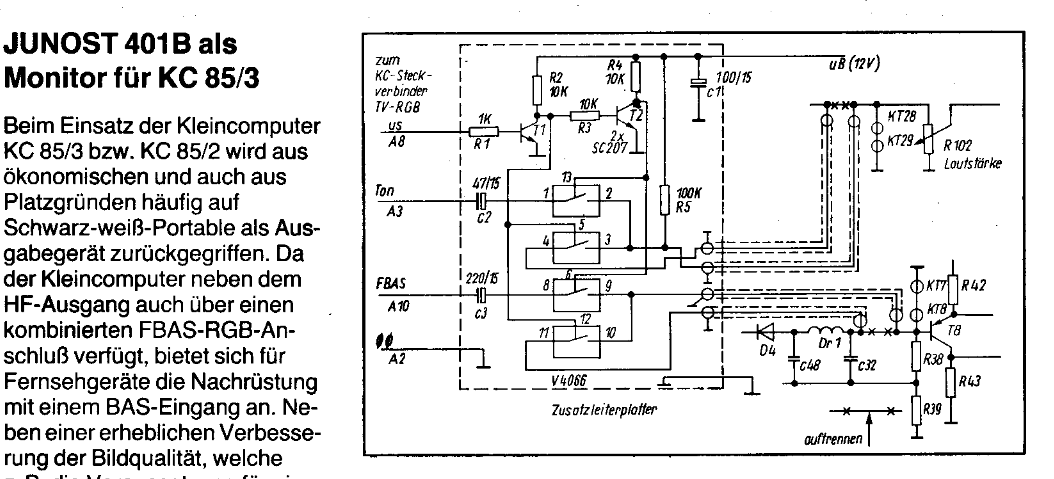


Bild 1 BAS-Ton-Anschluß im Junost 401B

Teleskopantenne erhielt einen Antennenstecker und kann jetzt über die Antennenbuchse betrieben werden. Als Verbindung zum Computer kommt ein 4adriges abgeschirmtes Kabel mit 26poligem direktem Steckverbinder zum Einsatz.
Bei ausgeschaltetem bzw. nicht angeschlossenem Computer wird das Fernsehgerät normal verwendet. Wird der KC eingeschaltet, so erfolgt automatisch die Umschaltung auf BAS-Eingang. Die vorgestellte Lösung ist auf andere Schwarzweiß-Fernsehgeräte mit galvanischer Netztrennung (!) übertragbar. Als Einkoppelpunkt für das BAS-Signal sollte die erste Verstärkerstufe nach dem Demodulator dienen. Das Tonsignal wird zweckmäßigerweise vor dem Lautstärkerregler eingespeist. Aber auch bei Farbfernsehgeräten bietet ein FBAS-Eingang, wie er hier vorgestellt wurde, eine Qualitätsverbesserung gegenüber dem HF-Eingang, wobei natürlich die volle RGB-Qualität nicht erreicht werden kann.

Klaus-Dieter Kirves

Literatur
/1/ Schlenzig, St.; Schlenzig, K.: Ein Textsystem für KC 85/2 Radio, Ferns., Elektron. 35 (1986) 8, S. 501-502
/2/ Faulenbach, U.: FS-Empfänger „Junost 402B“ ein Monitor. Funkamateur 34 (1985) 12, S. 619

Ladeadressenanzeige für KC 85/2

Das Programm CLIB zeigt die auf einer Kassette vorhandenen Dateien und ihre Ladeadressen an. Bei Basic-Programmen werden als Ladeadressen die ersten Bytes des Programms angezeigt. Der Aufruf erfolgt mit CLIB zahl, wobei „zahl“ die Anzahl der anzuzeigenden Dateien ist. „zahl“ muß zwischen 1 und 255 liegen und sollte nicht größer sein als die tatsächlich vorhandene Anzahl. Fehlt „zahl“, dann wird der Durchlauf von 256 Dateien erwartet. Es ist möglich, die Kassette zu wechseln oder vorwärts und rückwärts zu spulen. Das Originalprogramm steht ab Adresse 70H.

```
7F 7F 43 4C 49 42 01 45
C5 DC 03 F0 0A DD 7E 02
3D 20 F6 DD 6E 05 DD 66
06 06 0B 7E CD 03 F0 00
23 10 F8 CD 03 F0 2B 06
06 23 10 FD 5E 23 56 23
D5 5E 23 56 E1 CD 03 F0
1B CD 03 F0 2C C1 10 C8
CD 03 F0 0B C9
```

M. Lennartz

Anschluß von Kassettenmagnetbandgeräten

Trotz ausführlicher Hinweise in der Bedienungsanleitung der Kleincomputer KC 85/1 und KC 87 zur Arbeit mit dem Kassettenmagnetbandgerät (KMBG) treten immer wieder Fehlfunktionen beim Einlegen bzw. Aufzeichnen auf, für deren Behebung an dieser Stelle einige Hinweise gegeben werden sollen.
① Es sind nur Kassettenmagnetbandgeräte einsetzbar, die der TGL 28200/05 hinsichtlich Kontaktbelegung und Spannungspegel entsprechen und eine obere Grenzfrequenz ≥ 8 kHz besitzen.
② Unbedingt Diodenkabel, keine Überspielkabel verwenden.
③ Das Anschlußkabel nicht versehentlich in eine Spielhebelbuchse stecken. Damit wird die Tastaturabfrage blockiert. Hartnäckiges Betätigen der jetzt wirkungslosen Tasten kann zur Zerstörung des PIO-Schaltkreises U 855 führen.
④ Beim Aufzeichnen von Programmen auf Magnetband keinesfalls die Aussteuerautomtik benutzen, sondern durch Handaussteuerung eine leichte Übersteuerung einstellen (in alten Bedienungsanleitungen steht dazu ein falscher Hinweis!).

Dr. G. Kleinmichel

Der Computer als intelligentes Arbeitsmittel

Anläßlich des Jubiläums der Gründung des Organisations- und Rechenzentrums der Martin-Luther-Universität Halle-Wittenberg vor 25 Jahren fand vom 24. bis 26. 9. 1986 eine repräsentative Konferenz zum Thema *Der Computer als intelligentes Arbeitsmittel* statt. Mehr als 300 Gäste aus dem In- und Ausland, mehr als 50 Vorträge in drei Sektionen bzw. vor dem Plenum, eine ausgezeichnete Organisation und eine diskussions- und wissenschaftsträchtige Tagungsatmosphäre bestimmten den äußeren Rahmen der Veranstaltung.

Sowohl die im folgenden genannten Plenarvorträge als auch die einzelnen Sektionen stellten die Informatik sowohl als Wissenschaftsdisziplin als auch in ihrer Praxisrelevanz und volkswirtschaftlichen Bedeutung umfassend dar, wobei eine besondere Betonung der Entwicklung der Künstlichen Intelligenz zu bemerken war.

Plenarvorträge

- Lernfähige Systeme (*Reichl*)
- Spezielle Maschinen und Parallelalgorithmen für Künstliche Intelligenz (*Mikloško*)
- Evolution auf dem Gebiet der problemorientierten Programmiersprachen (*Stiller*)
- Überlegungen zur Rolle von Erkenntnismitteln bei der Konstituierung wissenschaftlicher Gegenstände (*Langhammer*)
- Stand und Entwicklungstendenzen der arbeitsplatzbezogenen Mikrorechentechnik (*Bernstein*)
- Anwendungen der Künstlichen Intelligenz in CAD/CAM-Systemen (*Posthoff*)

Sektion 1 beschäftigte sich umfassend mit theoretischen und praktischen (!) Problemen der Künstlichen Intelligenz wie Algorithmisches Lernen, Automatische Sprachverarbeitung, Wissensdarstellung, Expertensysteme, PROLOG; Schwerpunkt der Sektion 2 bildeten Datenbanken und ihre Anwendung; Sektion 3 enthielt viele stark anwendungsorientierte Ergebnisse bis hin zu Anwendungen in der Medizin – Ansätze für eine Medizinische Informatik sind hier zu sehen!

Es zeigte sich, daß die Entwicklung der Künstlichen Intelligenz in Theorie und Praxis Fortschritte macht und daß man wohl bald zu Tagungen übergehen kann (muß), die auf Teilgebiete konzentriert sind. Die ganze Breite der Informatik ist mit der zur Verfügung stehenden Zeit kaum noch zu bewältigen.

Abschließend sei gewürdigt, daß mit dieser Tagung auch das 25jährige Dienstjubiläum von Prof. J. Krötenheerdt als Direktor des ORZ der Martin-Luther-Universität begangen wurde.

Prof. Dr. Ch. Posthoff



Applikation Mikroelektronik – Stand und Tendenzen

Im Mittelpunkt der 5. Fachtagung *Applikation Mikroelektronik – Stand und Tendenzen*, die am 27. und 28. November 1986 in Dresden stattfand, standen wesentliche Arbeitsergebnisse, die bei der Realisierung des KDT-Initiativprogramms des Fachverbandes Elektrotechnik vom Mai 1986 *Entwicklung der Elektrotechnik und Elektronik unter den Bedingungen der intensiven Entwicklung der Applikation der Mikroelektronik* bisher erzielt werden konnten.

Mit 62 Vorträgen von Fachkollegen aus der DDR und 4 Vorträgen von Gästen aus der UdSSR, ČSSR und der VR Polen wurde in 5 Arbeitssektionen und im Rahmen einer Posterdiskussion ein praxisbezogener Informations- und Erfahrungsaustausch zu den erreichten Ergebnissen bei der Entwicklung und Anwendung mikroelektronischer Bauelemente und Systemlösungen geführt.

So wurde z. B. in der Sektion *Erfahrungen mit dem Einsatz von Kleincomputern und Entwicklungssytemen* über die Fortschritte berichtet, die auf diesem Gebiet seit der 4. Fachtagung im Jahr 1984 erreicht werden konnten. Mit Interesse wurden auch die Ausführungen aus der Ingenieurschule Velten-Hohenschöpping zum Einsatz von Mikrorechnern in der Aus- und Weiterbildung, die seit 1980 auf dem Gebiet der Informatikausbildung und Spezialausbildung mit dem MC 80, KC 85/2 und /3 sowie im Computerclub praktiziert wird, zur Kenntnis genommen. Aus dem Kombinat VEB EAW „Friedrich Ebert" Berlin wurde das moderne Programmier- und Entwicklungssystem P 8000 vorgestellt und über bisher vorliegende Erfahrungen berichtet.

Eingehend wurde über die Einsatzerfahrungen und Hauptanwendungen des im VEB Elektronik Gera entwickelten Mikrocomputer MC 80.30 informiert und auf die notwendige Komplettierung des bisherigen Angebots an Baugruppen (MC 80 – Beistelleinheit; ergänzende Baugruppen: Digital-Eingabe und -Ausgabe-Baugruppe, Analog-Digital-Umsetzer u. a.) hingewiesen. Hervorzuheben sind die Bemühungen, eine durchgängige Erhöhung der Zuverlässigkeit des elektronischen Massenspeichers zu erreichen, zu dem im Betrieb weitere Untersuchungen angestellt werden.

Vorgestellt wurde die *Modulverwaltung des Kleincomputersystems KC 85* aus dem VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, das durch weitere Module und einen Aufsatz mit mehreren Modulsteckplätzen in den nächsten Jahren weiter ausgebaut werden soll. Im System KC 85 sind alle Module mit einer während des Betriebes programmierbaren Modulsteuerung ausgerüstet. Damit ist eine optimale Anpassung des Computers an den entsprechenden Anwendungsfall möglich.

Um die Funktionsanpassung zu verbessern und das Leistungsvermögen zu erhöhen, wurde im VEB Kombinat Robotron zum Grundgerät des KC 85/1, das einmal als Heimcomputer konzipiert war, ein umfangreiches Sortiment an Hardware- und Softwareerweiterungen entwikkelt, über das im Vortrag informiert wurde. Die prinzipiellen Eigenschaften des Computers werden auch bei der Weiterentwicklung zum KC 87 beibehalten, so daß Erweiterungsbaugruppen, die ab 1987 verfügbar sind, dafür kompatibel sind.

Ergänzend zum ausgedruckten Programm wurde über die Nutzung der Software des PC 1715 zur Entwicklung von Anwenderlösungen berichtet. Dazu wurden das Betriebssystem BROS, JAMP und SCP (als Hauptbetriebssystem) und die übrigen Softwarekomponenten (Systemsoftware und Anwendersoftware) erläutert, die auf dem Computer nur unter Steuerung des jeweiligen Betriebssystems lauffähig sind. Hinsichtlich der Anwendersoftware wurde dabei auf Fragen eingegangen, die sich aus der Nutzung von individuellen Programmiersprachen oder speziellen Applikationsprogrammen (z. B. REDABAS) oder auch bei Standardsoftware ergeben.

Dem wachsenden Interesse nach Kleinrechentechnik wurde durch Einordnung eines zusätzlichen Vortrages *Hard- und Software des FKC 16 – ein Computer für die Kleinrationalisierung, Ausbildung und Lehre* in das Programm Rechnung getragen. Der Rechner entstand im Ingenieurbetrieb für die Anwendung der Mikroelektronik Frankfurt (Oder) und läßt sich gut in die vorliegenden DDR-Rechnerkonzepte einordnen. Er wurde zwar als Lerncomputer konzipiert, ist aber auch als Ingenieur-Arbeitsplatzrechner verwendbar und unterbietet in dieser Hinsicht bekannte Lösungen im Kosten-/Leistungsverhältnis recht beachtlich.

Die Vorstellung der Kleinrechner und deren praktische Demonstration ergänzten die Vorträge dieser Sektion. Damit konnte die wissenschaftlich-technische Leistungsfähigkeit der Computer allen interessierten Tagungsteilnehmern vorgeführt und zu einem intensiven Erfahrungsaustausch genutzt werden.

Die Vorträge und Diskussionen in den übrigen Arbeitssektionen

- moderne mikroelektronische Bauelemente und ihre Anwendungsmöglichkeiten
- rechnergestützte Vorbereitung und Durchführung der Produktion
- modulare und flexible Automatisierungslösungen
- Posterdiskussion zu mikroelektronischen Rationalisierungslösungen

ließen die bisher erreichten Ergebnisse der *Applikation Mikroelektronik* erkennen, machten aber auch die noch vor uns stehenden Aufgaben deutlich.

Die Diskussionsrunde während des diesjährigen Podiumsgesprächs *Erfahrungen aus der Arbeit von Computerclubs und Mikroelektronik-Arbeitsgemeinschaften* unterstrich nachdrücklich den erreichten Stand und die notwendigen weiteren Anstrengungen auch der Fachgremien der KDT, um die Entwicklung und Anwendung der Mikroelektronik weiter zu beschleunigen und umfassend durchzusetzen.

Dr. R. Schneider



Bericht

Electronica '86

Vom 11. bis 15. November 1986 fand in München (BRD) die 11. Internationale Fachmesse für Bauelemente und Baugruppen der Elektronik „Electronica '86" statt, zu der auch vier wissenschaftliche Veranstaltungen durchgeführt wurden, darunter der 12. Internationale Kongreß Mikroelektronik unter dem Leitthema *Mikroelektronik – Ausblick auf Technik und Märkte.*
2 400 Aussteller aus 36 Ländern waren in den 26 Ausstellungshallen auf 105 000 m^2 vertreten. Vor allem die westeuropäischen Bauelementeproduzenten sowie die Aussteller aus Übersee und Japan zeigten mit ihren neuesten Exponaten den erreichten wissenschaftlich-technischen Stand und die technologische Reife der Produkte in den Sektionen Halbleitererzeugnisse, optoelektronische Bauelemente, Röhren; passive Bauelemente; Baugruppen in Hybrid- und Leiterplattentechnik, Stromquellen; elektromechanische Elemente und mechanische Geräteteile; automatisierte Prüfeinrichtungen sowie Entwurfshilfsmittel für Mikro- und Makroelektronik (CAE und CAD).
Die DDR war durch die Außenhandelsbetriebe Elektronik Export-Import, Heimelektric und Robotron mit entsprechendem Fachprofil vertreten.
Der Stand der Produktion fortgeschrittener höchstintegrierter Schaltungen wurde durch die 2-Mikrometer-Technologien, z. B. beim 256-KBit-DRAM dokumentiert. Mit dem Entwicklungsabschluß des 1-Megabit-DRAM wurde die Produktion im 1 ... 1,2-μm-Niveau vorbereitet. Bei diesem Integrationsgrad von VLSI-Schaltkreisen dominierte die CMOS-Technologie. Texas Instruments z. B. verkleinerte die Zellengröße ihrer Speicher von 54 μm^2 beim 256-K-DRAM über 21,2 μm^2 beim 1-M-DRAM auf 8,9 μm^2 für den 4-M-DRAM durch „Vergraben" des bisher planaren Transfergatters in die Kreuzpunktzelle. Die Produktion des 4-M-DRAM soll 1988 beginnen. Mit ihrer EPIC-Technologie, einer CMOS-Variante, fertigt Texas Instruments auch Hochgeschwindigkeitslogikelemente. Das belegt anschaulich den Einfluß physikalischer Festkörpertechnologien auf die Erzielung progressiver Bauelementeeigenschaften.
Mikroprozessoren mit 32 Bit Verarbeitungsbreite wie der 80386 von Intel, 68020 von Motorola, V70 von NEC, Clipper von Fairchild u. a., die eine Taktrate bis ca. 16 MHz und eine durchschnittliche Leistung von 3–4 Millionen Instruktionen pro Sekunde (MIPS) aufweisen, demonstrieren den Fortschritt auf diesem Gebiet. Für 1986 wurde ein Umsatz von 2 Millionen Stück erwartet. Hauptanwendungsgebiete sind Hochtechnologieprodukte wie Computergrafik, Arbeitsstationen für CAD/CAM/CAE, Geräte zur Bild- und Sprachverarbeitung, Parallelverarbeitungs-, Mehrbenutzer-, Echtzeit- und andere Rechnersysteme; künftig Expertensysteme sowie die Datenverwaltung und -verdichtung in Kommunikationsnetzen.
Ein hohes Tempo des wissenschaftlich-technischen Fortschritts ist bei digitalen Signalprozessoren zu verzeichnen, spezialisierten Mikroprozessoren, vorwiegend in Havard-Architektur, die viele sequentielle Operationen in extrem kurzer Zeit auszuführen ermöglichen. Anwendungen sind in der Sprachverarbeitung, Bildverarbeitung und der Nachrichtentechnik vorgesehen. Als künftiger Absatzschwerpunkt wird die Videosignalverarbeitung betrachtet, für die das heute realisierbare Technologieniveau noch nicht ausreicht.
Die stürmische Entwicklung von anwendungsspezifischen integrierten Schaltungen, sogenannten ASICs, verdient besondere Aufmerksamkeit. Hier werden jährliche Steigerungsraten der Produktion von 30–40 % vorhergesagt, und 1995 wird ein Anteil von 25–30 % am gesamten Halbleitermarkt für diese Erzeugnisse erwartet. Dieser Entwicklung auf der Basis von programmierbaren Logikbauelementen, Gate-Arrays, Standardzellen und vollkundenspezifischen Schaltungen wird eine hohe Bedeutung bei der breiten Nutzung von VLSI-Technologien beigemessen. Diese Art des Entwurfs integrierter Schaltkreise trägt entscheidend dazu bei, die Zusammenarbeit von Bauelementehersteller und -anwender auf immer höherem Technologieniveau enger zu gestalten. Entsprechend ausgebaut wird hier das Angebot an Beratungen, Schulungen, Entwurfszentren und CAE-Werkzeugen für Anwender. Neue Anwendungsgebiete werden durch funktionelle Erweiterungen, die Erhöhung des Integrationsgrades und leistungsfähigere Technologien erschlossen. So wurden Standardzellenentwurfssysteme vorgestellt, die Zellen variabler Größen bis zu kompletten Mikroprozessoren enthalten, und es beginnt der Einsatz von Silizium-Compilern für die Generierung von Funktionsblöcken. Intel gibt z. B. an, daß ein Drittel der Struktur seines 32-Bit-Prozessors 80386 Standardzellen-Reihenstrukturen aufweist. Die dazu genutzten Entwurfsbibliotheken und CAE-Werkzeuge stellt diese Firma jetzt auch ihren ASIC-Kunden zur Verfügung. In ca. 2 Jahren sollen derartige Bauelemente Integrationsgrade von etwa 100 000 Gattern erreichen. Bei der Weiterentwicklung der Software für Entwurf und Test von anwenderspezifischen Schaltkreisen werden Fragen der Testbarkeit als Hauptelement des Entwurfsprozesses berücksichtigt; es wird Zusatzlogik für einen testfreundlichen Entwurf integriert.
Intel und Altera stellten die modulare Hardwareentwicklung von programmierbaren Logikelementen vor und schulten mit einem Entwicklungssystem, das auf einem leistungsfähigen Personalcomputer abläuft. Die gezeigten EPLD-Bauelemente enthalten EPROM-Zellen, die durch ultraviolettes Licht gelöscht und dann wieder programmiert werden können. Diese Technik ist für Anwender in der Industrieautomatisierung interessant, die selbst mikroelektronische Bauelemente entwerfen und herstellen wollen.
In der Nachrichtentechnik wurden mikroelektronische Bauelemente für die Digitalisierung als Grundvoraussetzung zur Realisierung eines ISDN-Konzeptes für die Sprach-, Daten-, Text- und Bildkommunikation herausgestellt. Die führenden Halbleiter- und Systemhersteller zeigten dazu ihre Entwicklungsergebnisse, insbesondere das Sortiment für den digitalen ISDN-Teilnehmer und die zentralenseitige Anschaltung. Die Schaltkreise wandeln die Signale aus den unterschiedlichen Informationsquellen in einheitlich codierte Impulsströme, die in den Vermittlungs- und Übertragungseinrichtungen verarbeitet werden. Zu den Schaltkreisen, die universell in einem ISDN einsetzbar sein sollen, gehören z. B. Bauelemente zur Anschaltung analoger Endgeräte an das digitale Netz (IOM-Interface von Siemens) oder Schaltkreise zur Realisierung der Funktionen Kodieren/Dekodieren/Filtern und Zeitlagenzuordnung (29 C53 und 29 C28 von Intel). ISDN-Schaltkreise erfordern das Integrationsniveau des 1-M-DRAM, für die U-Schnittstelle dazu analogfähig, und stehen als Beispiel für höchstintegrierte ASICs.
Zur Electronica '86 wurde eine breite Palette an Meßtechnik ausgestellt mit dem Schwerpunkt der automatisierten VLSI-Tester der Firmen aus Japan und den USA. Derartige Geräte mit 256 Ein-/Ausgangskanälen, Datenraten von 40 bis 100 MHz, aufgebaut mit Hochgeschwindigkeitsschaltkreisen und -signalprozessoren sowie Bauelementen in ECL bzw. GaAs-Technik, sind eine Grundvoraussetzung für die Produktion höchstintegrierter Schaltkreise und werden entsprechend deren Technologiefortschritt weiter entwikkelt. Automatische Tester haben z. Z. etwa ein Viertel der gesamten elektronischen Meßtechnik erobert, wobei der Einsatz zum Bauelementetest ca. 50 %, der Leiterplattentest ein Drittel und die Prüfung von Verbindungen und unbestückten Leiterplatten ca. 10 % des Wertvolumens ausmachen. Als Beispiel sei das System LT 1 000 von Tektronix genannt, das für die Testung kundenspezifischer Schaltkreise in CMOS-Technologie bei der Herstellung und im Wareneingang entwickelt wurde. Es ermöglicht über ein Ethernet-Netzwerk den Anschluß an CAE-Systeme. Mit Hilfe einer „künstlichen Intelligenz"-Software wurde die Entwicklungszeit von Testprogrammen von Tagen auf Stunden reduziert. Als bedeutungsvoll muß auch der zweitägige 12. Internationale Kongreß „Mikroelektronik" angesehen werden. Dieser Kongreß bot Einblick in marktorientierte Fragestellungen und stellte Ergebnisse sowie künftige Aufgaben der wissenschaftlich-technischen Entwicklung dar. Weitere wissenschaftlich-technische Veranstaltungen waren der Makroelektronik, der Sensorik und der Qualitätssicherung gewidmet. Die Vortragsreihen beschäftigten sich mit neuen Bauelementen und Schaltungskonzepten der Leistungselektronik, lokalen Sensorsystemen mit Mitteln der Digitaltechnik und dem gemeinsamen Bemühen von Bauelemente- und Geräteherstellern um eine hohe Betriebssicherheit elektronischer Geräte, besonders beim Übergang zu neuen, effektiveren Bestükkungstechnologien von Leiterplatten wie der Oberflächenmontage (SMT).

Ha.

Mikroelektronik und deren Bauelemente

Lexikonreihe: 1000 Begriffe für den Praktiker
Herausgeber: K.-P. Scholz, VEB Verlag Technik Berlin 1986, 1. Auflage, 270 S.; 2. unveränderte Auflage 1987

Das Buch beginnt mit einer halben Seite einführender Hinweise und faßt dann mit vielen wichtigen Begriffen den grundsätzlichen Grundwissensschatz zu den meisten Problemkreisen der modernen Mikroelektronik alphabetisch geordnet zusammen. Die Begriffe sind gut ausgewählt, systematisch und prägnant beschrieben und logisch geordnet. Natürlich bleiben dabei auch manche Wünsche offen. Das sind aber zumeist allgemein noch unklar eingeführte Begriffe, wie z. B. Interrupt-Befehl (eben kein richtiger Befehl), seltener grundsätzliche Kernbegriffe wie z. B. Schaltung (besteht aus Schaltelementen; (ideale elektrische) Verbindungen und R, L, C, T, ... (in Form von deren Schaltsymbolen, die wiederum durch entsprechende Ersatzschaltungen ersetzt werden können)).
Insgesamt kann dem Leser das Buch warm empfohlen werden. Er wird gern und mit Gewinn nachschlagen und blättern.
Prof. Dr. D. Eckhard

Handbuch TTL- und CMOS-Schaltkreise

von E. Kühn, VEB Verlag Technik Berlin, 2. Auflage 1986, 412 S.

TTL-Schaltungen haben in den 60er Jahren den Siegeszug Integrierter Schaltungen (IS), der Digitaltechnik und der (Digital-) Rechentechnik begründet. Sie haben die Schaltkreisreihen als universelle Baukästen für den Aufbau beliebiger (komplexer digitaler) Schaltungen aus Integrierten Schaltungen vorgeprägt mit den damals beherrschbaren Integrationsgraden SSI und MSI. Die TTL-Schaltkreis-Reihen sind in ihrem Typenspektrum ständig erweitert und um neue modernere Reihen ergänzt worden. Sie geben die Standard-Anschluß/Interface-Normen auch für fast alle LSI-, VLSI- sowie Spezial-IS vor. In den letzten Jahren wurden CMOS-Reihen als Ablösemöglichkeiten von TTL-Reihen eingeführt. In diesem Sinne ist das Buch ein guter Überblick zum Gesamtproblem, zum Typenspektrum der Reihen, zu Aufbau, Funktion und Anwendung der TTL-und CMOS-Technik. Es faßt sehr kompakt und doch leicht faßlich sehr viel Wissenswertes zusammen und hilft dem Praktiker entscheidend. Die Abschnitte geben einen Überblick über den Inhalt: Systematik der Schaltkreise, Gatterschaltkreise, Flip-Flop-Schaltkreise, Übersichten: TTL- und CMOS-Schaltkreise, statische Eigenschaften von TTL-Schaltkreisen, dynamische Eigenschaften von TTL- und CMOS-Schaltkreisen, Störeinflüsse bei TTL-Schaltkreisen, CMOS-Schaltkreise, High-speed-CMOS-Schaltkreise, Impulsgeneratoren, spezielle Schaltungsfunktionen, Zähler und Teiler, Schieberegister, Dekodiernetzwerke, Multiplexer, Addierer, programmierbare Schaltkreise, Stromtreiber, Leistungstreiber und Empfänger, langsame störsichere Logikbaureihen; Anhänge: TTL- und CMOS-Schaltkreise der Baureihen 74 (Typenspektrum), CMOS-Schaltkreise der Baureihen 4000 (Typenspektrum), Vergleichsliste Digitalschaltkreise.
Das Buch gibt einen erschöpfenden Überblick zu TTL- und angepaßten CMOS-Reihen bis zum Erscheinungsjahr 1982 sowie zu langsamen/störsicheren Reihen der Automatisierung. Die schnellen ECL-Reihen der Hochleistungs-Reichen- und Kommunikationstechnik sind nicht behandelt worden.
Das Buch ist andererseits auch in den Begriffschwächen unserer Zeit festgelegt. So beginnt der erste Satz mit „Ein Schaltkreis besteht aus einer Anordnung von Bauelementekomponenten, ...“ – das ist schon ein Monster! Das Zitat beginnt und endet mit Eindeutschungsmißgeburten: Schaltkreis statt richtig Schaltung ggf. integrierte Schaltung infolge einer integrierten Herstellung, Bauelementekomponenten statt richtig Schaltelementen oder ggf. Schaltungselementen. Doch ohne solche Monster wäre alles zu einsichtig.
Insgesamt muß man zu diesem Buch nicht mehr besonders zuraten, es hat sich ja bereits als unentbehrliches Nachschlagewerk durchgesetzt!
Prof. Dr. D. Eckhardt

Flexible Automatisierung

Autorenkollektiv, Verlag die Wirtschaft Berlin 1986, 96 S.

Die Schrift des Verlages Die Wirtschaft mit dem Titel *Flexible Automatisierung* und dem Untertitel *Schlüseltechnologie für höhere Produktivität und Effektivität* – geschrieben von den in Planung und Wirtschaft hochprofilierten Autoren G. Proft, K.-P. Dittmar, H. Gerke, R. Winter, E. Adam, H. Berteit und H. Freimüller – erweist sich bei der Lektüre als hochaktueller Ratgeber und Agitator bei der Durchsetzung der komplexen Rationalisierung und Automatisierung insbesondere in der fertigungstechnischen Industrie. Dazu werden sowohl konkrete, fast rezeptive Informationen gegeben, als auch verallgemeinerte, beliebig übertragbare Schlußfolgerungen für die Durchsetzung der Beschlüsse des XI. Parteitages gezogen.
Die flexible Automatisierung wird völlig zurecht über die Fertigungsvorbereitung und die Fertigungssteuerung hinaus auf die „durchgängig rechnerintegrierte Produktion“ (CIM, Computer Integrated Manufacturing) qualitativ gehoben, um so die Herausbildung der mindestens in einer Schicht bedienfreien Produktion (automatisierte Fabrik) als entscheidendes Ziel der Entwicklung unserer Produktivkräfte zu begründen. In dieser Logik werden sowohl parteistrategische, leitungsorganisatorische, objektbezogene technische als auch soziologische und humanistische Fragen prinzipiell deklarativ behandelt. Die reichen Erfahrungen, die von den Autoren insgesamt vermittelt werden, sind ohne Zweifel ein Fundus für politische, ökonomische und technische Leiter gleichermaßen wie für Ingenieure und Arbeiter, die selbst einen solchen komplexen Prozeß bewältigen wollen.
Prof. Dr. M. Roth

Strategie der Haie

von Heerke Hummel. Urania-Verlag Leipzig. Jena Berlin, 1986, 159 S., Reihe klartext

Einen Blick hinter die Kulissen des Hochkapitals gestattet die vorliegende Broschüre. Gekonnt verbindet der Autor Authentisches – herausgearbeitet aus zahlreichen Quellen – mit einem fiktiven Erzählhintergrund. Dadurch ist die Publikation spannend und unterhaltsam zugleich zu lesen. Besonders gelungen sind die Kapitel über die Konzerne IMB-„Big-Blue“ und Siemens; zeigen sie doch detailliert auf, wie sich die beiden Unternehmen zu Giganten entwickeln konnten. Die geschilderten Fakten sprechen dabei für sich. Sie ergeben ein realistisches Bild über das Gebahren internationaler Konzerne, wo jedes Mittel recht ist, um Profit zu erzielen. Die vielen Karikaturen bilden eine gute Ergänzung zum Text.
I. Paszkowsky

VENUS-Entwurf von VLSI-Schaltungen

von E. Hörbst, M. Nett, H. Schwärtzel, Springer-Verlag, Berlin-Heidelberg-New York-Tokyo 1986, 336 S.

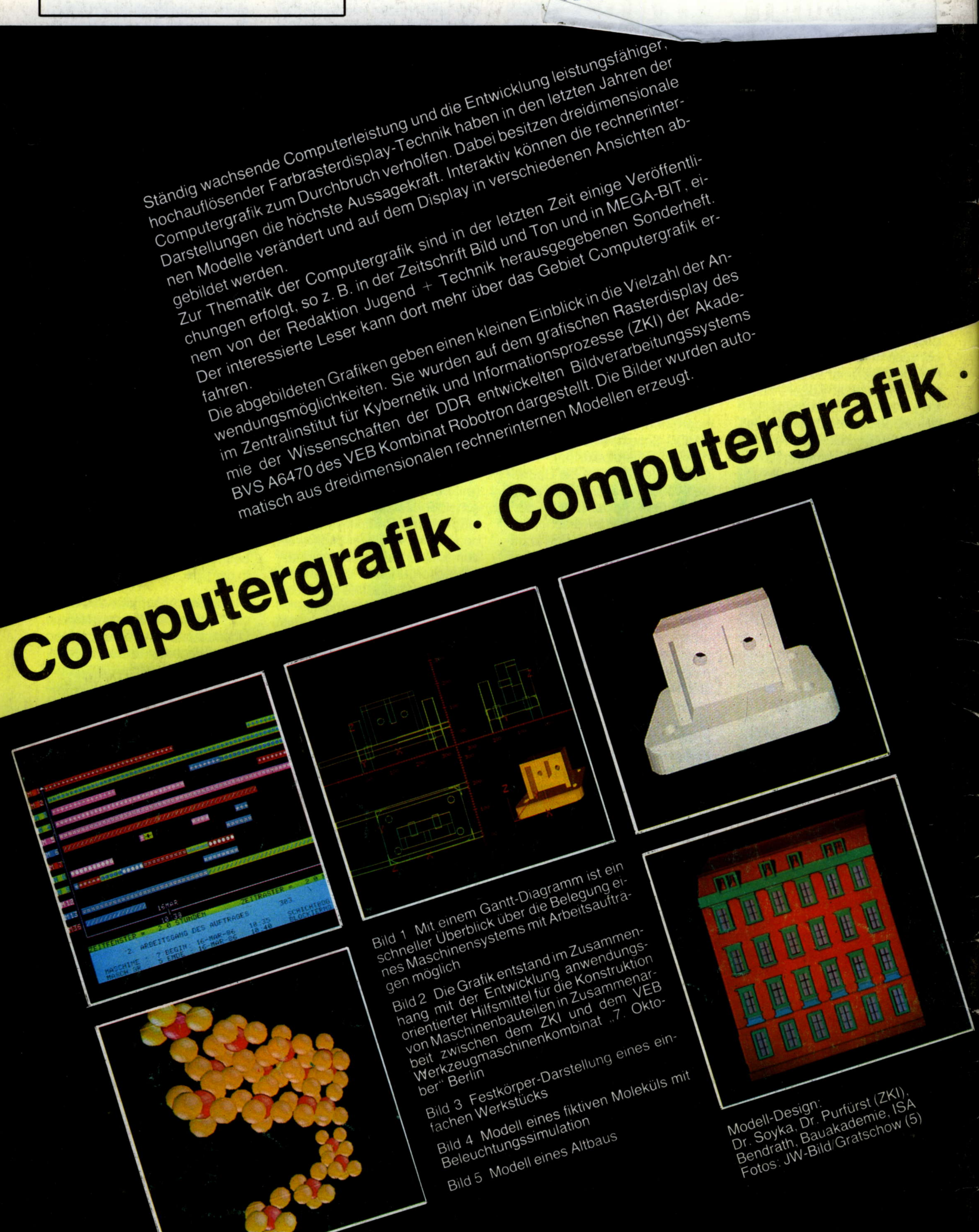
Für alle Technikbereiche ist die Mikroelektronik der entscheidende Innovationsmotor. Das gilt anerkanntermaßen seit einigen Jahren besonders für Mikroprozessor-Softwarelösungen. Jetzt zeichnet sich aber ab, daß auch Hardwarelösungen auf der Grundlage von VLSI-Schaltkreisen als sogenannte ASIC (application specific integrated circuits) zunehmend wirksam werden. Die Trendaussagen weisen auf ein überdimensionales Wachstum hin.
Das vorliegende Buch ist diesem Problemkreis gewidmet. Grundlage ist das gleichnamige Schaltkreis-CAD-System von Siemens. Dieses System befindet sich seit längerer Zeit in der praktischen Nutzung.

Die sieben Abschnitte des Buches geben einen sehr guten Überlick zu allen derzeit praktisch relevanten Problemen:

- Einführung in die Designtechnik für integrierte Schaltungen;
- Einführung in die Halbleitertechnologie für integrierte Schaltungen;
- Layoutdesignmethoden;
- Prüftechnische Konzepte;
- Zellen und Bibliotheken;
- Einsatz des Entwurfssytems VENUS;
- Ausblick.

Das Buch enthält zahlreiche praktische Beispiele in der Gegenüberstellung von Hard- und Softwarelösungen.
Im recht kurzen letzten Abschnitt wird auf die Breitenanwendung und die Integration mit anderen (z. B. globaleren) Entwurfssystemen hingewiesen. Siliconcompilerlösungen werden im Prinzip im gesamten Inhalt methodisch vorbereitet, aber erst am Schluß explizit angesprochen.
Das Buch ist nicht nur für VENUS-Nutzer von Interesse.
Prof. Dr. D. Eckhardt

ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 4
S. 97-128



Computergrafik · Computergrafik ·

Ständig wachsende Computerleistung und die Entwicklung leistungsfähiger, hochauflösender Farbrasterdisplay-Technik haben in den letzten Jahren der Computergrafik zum Durchbruch verholfen. Dabei besitzen dreidimensionale Darstellungen die höchste Aussagekraft. Interaktiv können die rechnerinternen Modelle verändert und auf dem Display in verschiedenen Ansichten abgebildet werden.

Zur Thematik der Computergrafik sind in der letzten Zeit einige Veröffentlichungen erfolgt, so z. B. in der Zeitschrift Bild und Ton und in MEGA-BIT, einem von der Redaktion Jugend + Technik herausgegebenen Sonderheft. Der interessierte Leser kann dort mehr über das Gebiet Computergrafik erfahren.

Die abgebildeten Grafiken geben einen kleinen Einblick in die Vielzahl der Anwendungsmöglichkeiten. Sie wurden auf dem grafischen Rasterdisplay des im Zentralinstitut für Kybernetik und Informationsprozesse (ZKI) der Akademie der Wissenschaften der DDR entwickelten Bildverarbeitungssystems BVS A6470 des VEB Kombinat Robotron dargestellt. Die Bilder wurden automatisch aus dreidimensionalen rechnerinternen Modellen erzeugt.

Bild 1 Mit einem Gantt-Diagramm ist ein schneller Überblick über die Belegung eines Maschinensystems mit Arbeitsaufträgen möglich

Bild 2 Die Grafik entstand im Zusammenhang mit der Entwicklung anwendungsorientierter Hilfsmittel für die Konstruktion von Maschinenbauteilen in Zusammenarbeit zwischen dem ZKI und dem VEB Werkzeugmaschinenkombinat „7. Oktober" Berlin

Bild 3 Festkörper-Darstellung eines einfachen Werkstücks

Bild 4 Modell eines fiktiven Moleküls mit Beleuchtungssimulation

Bild 5 Modell eines Altbaus

Modell-Design:
Dr. Soyka, Dr. Purfürst (ZKI),
Bendrath, Bauakademie, ISA
Fotos: JW-Bild/Gratschow (5)

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK



Heft 5 · 1987

Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0233-2892

CAD
Koordinaten BE-Kennwerte
ATG
Testprogramm
ATE
Fehlerdaten
CAR
Daten für Entwurfskorrektur

Fertigungsorientierte Meß- und Prüftechnik – Stand und Tendenzen

Software für Mehrmikrorechnersysteme

Schneller 12-Bit-A/D-Wandler C 574 C

Bildverarbeitungssystem A 6471

Automatisierte Mikroskopbildanalyse

Für die Anwendung auf dem Gebiet der Mikroskopbildanalyse in der Medizin und Biologie wird das Softwarepaket **AMBA/R-MED** angeboten.

AMBA/R-MED ermöglicht die Diagnostik und Therapiekontrolle durch Analyse von Geweben, Zellen und Zellkernen sowie Zellorganellen.

Die erfaßten Parameter gestatten eine Quantifizierung von Zellpopulationen und Gewebeveränderungen.

Weiterhin ermöglicht AMBA/R-MED
- die Charakterisierung typischer Zellverbände
- die Quantifizierung von Faserstrukturen und
- die Analyse der Objektverteilung im mehrdimensionalen Merkmalsraum.

Alle Programme des Paketes AMBA/R-MED werden als Quellprogramme zur Verfügung gestellt. Das ermöglicht problemlos Ergänzungen und Erweiterungen der speziellen Aufgabenstellung.
Mit AMBA/R-MED steht somit ein Paket von Programmen zur Verfügung, das den Service des interaktiven Softwarebasissystems AMBA/R-MED nutzt und das als Grundlage für effektive Untersuchungen von Objekten dient, die reproduzierbar sind und subjektive Einflüsse ausschließt.

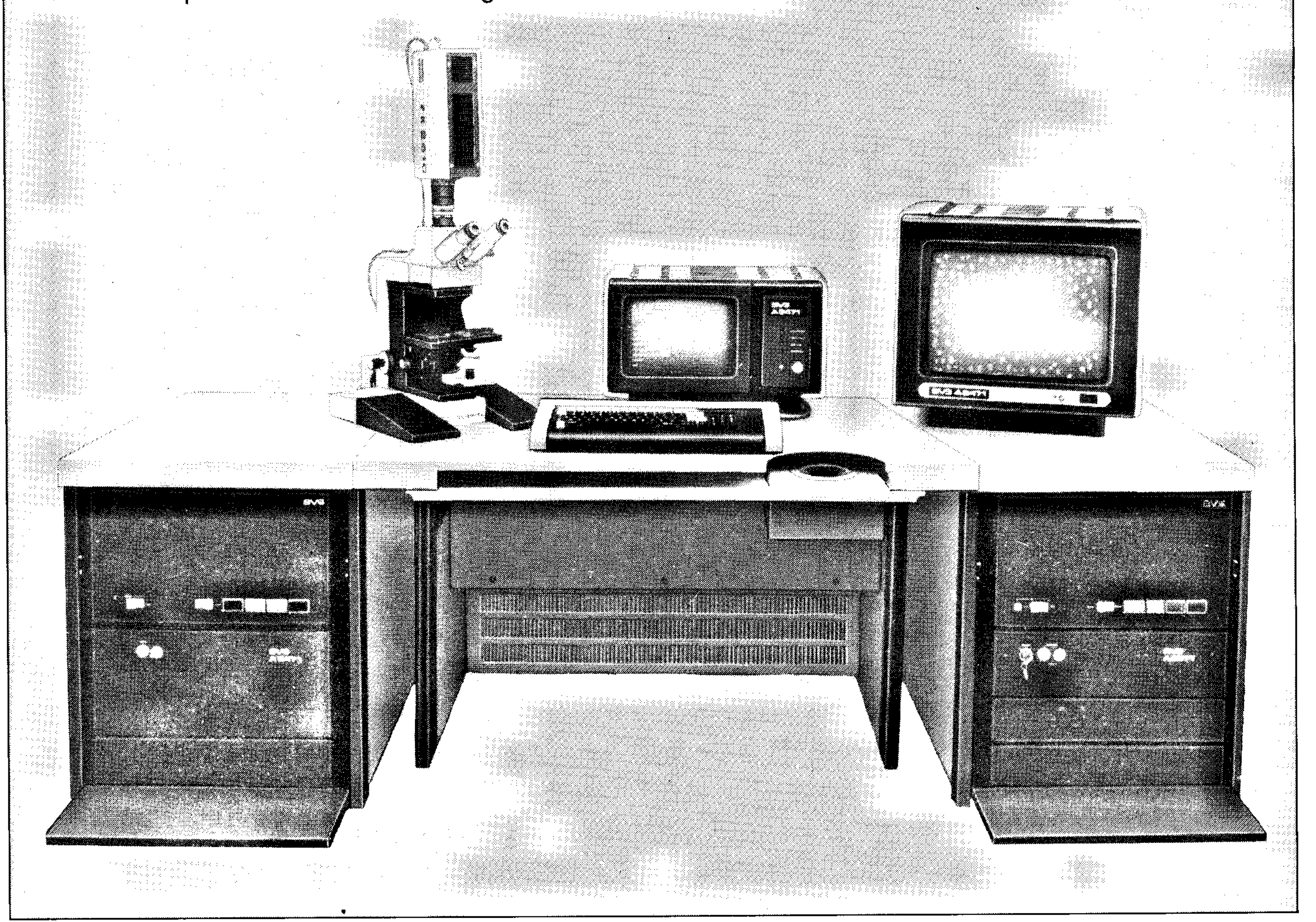
Die automatisierte Mikroskopbildanalyse mit AMBA/R-MED wird mit dem digitalen Bildverarbeitungssystem robotron A 6471 einschließlich Mikroskop und Videokamera durchgeführt.

robotron

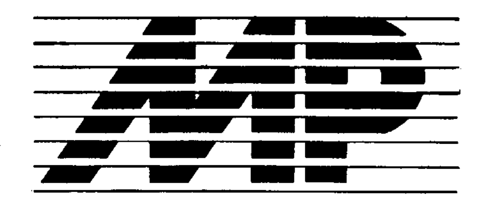
VEB Robotron-Vertrieb Berlin
Mohrenstraße 62
Berlin
DDR – 1080

Exporteur:
Robotron Export-Import
Volkseigener Außenhandelsbetrieb
der Deutschen Demokratischen Republik
Allee der Kosmonauten 24
Berlin
DDR – 1140

DEWAG LEIPZIG



Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0233–2892



Mikroprozessortechnik, Heft 5 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)

Titel: Keller

Beirat Dr. Ludwig Claßen, Prof. Dr. sc. Dietrich Eckhardt, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Jürgen Zaremba

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

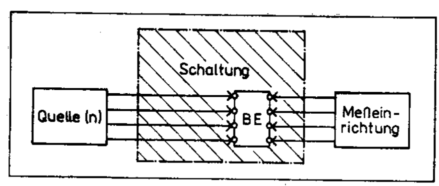
Redaktionsschluß: 26. März 1987

AN (EDV) 49837

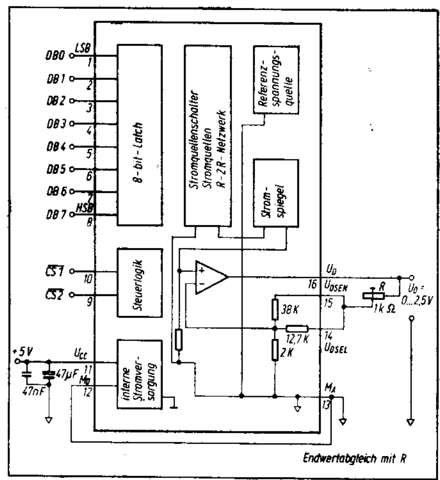
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

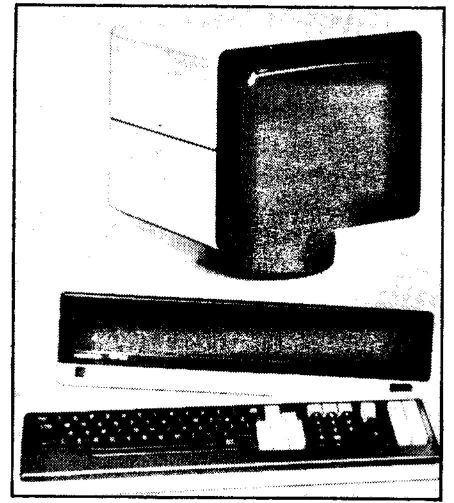
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010, und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Seite 131



Seite 140



Werkfoto

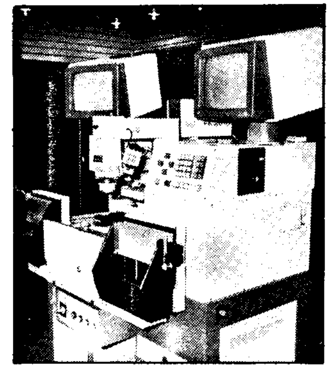
LFM '87

Unser Bericht von der Leipziger Frühjahrsmesse gibt einen Überblick über den wiederum stark vertretenen Bereich der Computertechnik und ihrer Bauelementebasis.
Für viele Anwender und für potentielle Nutzer von Interesse war zum Beispiel der weiterentwickelte PC 1715, der jetzt u. a. eine noch bessere Textverarbeitung erlaubt. Dazu dienen die Tastatur mit dem Textprozessor TP, mit entsprechenden Funktionstasten und in der deutschen Version mit Umlauten sowie ein spezielles Softwarepaket mit einer Fülle von Funktionen, unter anderem auch zur Arbeit mit Dateien (KOMBO-Druck).

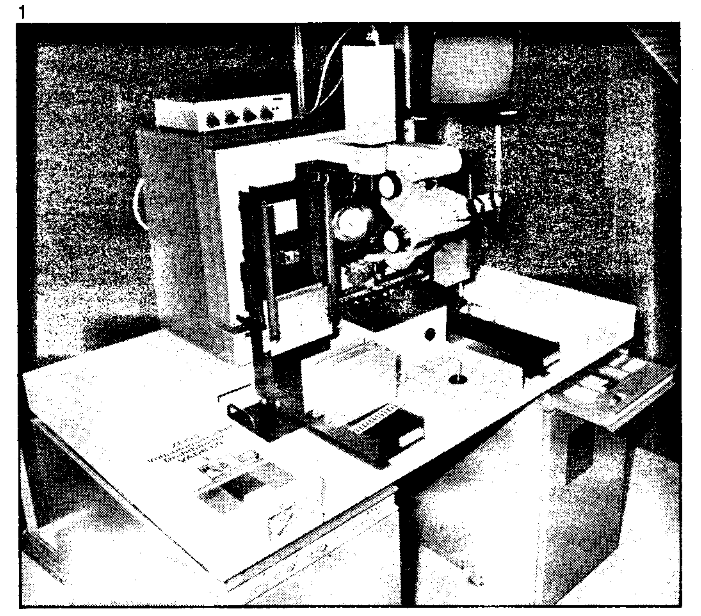
Inhalt

Technologische Ausrüstungen zur Chipherstellung

Geräte für die Hochtechnologien stellte das Kombinat VEB Carl Zeiss JENA auf der Leipziger Frühjahrsmesse 1987 vor. In einem Raum in Halle 14 war eine Gerätestrecke für die Chipherstellung zu sehen. Das erste Glied bildete dabei der automatische Vielfachsondentaster AVT 120. Mit ihm erfolgt die elektrische Prüfung der Schaltungen auf den Halbleiterscheiben nach dem Zyklus I. Es können statische und dynamische Tests an analogen und digitalen Schaltungen durchgeführt werden. Die Ein- und Ausgabe der Halbleiterscheiben – sie können einen Durchmesser von 76 bis 150 mm (3 bis 5 Zoll) besitzen – kann neben der automatischen, magazinierten Form auch manuell erfolgen. Die bauelementespezifischen Daten (Durchmesser, Dicke usw.) können wahlweise über Tasten oder durch ein austauschbares EPROM eingegeben werden. Durch einen Kreuztisch mit linearen Schrittantrieben werden in Verbindung mit dem dazugehörigen Stellsystem die Halbleiterscheiben in den 4 Koordinaten x, y, z und φ positioniert. Die Schritteinheit beträgt 5 μm; der Positionierfehler liegt bei ± 10 μm (bei 20°C). Als Steuerrechner fungiert ein K 1520.
MDB10 und MDB20 sind manuelle Drahtbonder. Der MDB20 ist ein Golddrahtbonder für die Thermosonic- und Thermokompressionskontaktierung. MDB10 ist dagegen ein Ultraschallbonder zur Kontaktierung von Aluminiumdrähten mit einer Stärke


2 *Fotos: Paszkowsky (2)*

von 17,5 μm bis 100 μm. Das Bonden mit Aluminiumdrähten hat nicht nur den Vorteil des billigeren Rohstoffes, es ist auch besser geeignet für Chips, die nur wenig thermisch belastet werden können.
Für beide Geräte werden als Hauptanwendungsgebiete Forschung und Entwicklung, Kleinserienfertigung und die Verwendung als Reparaturbonder in der Großserienfertigung angegeben.
In der Großserienfertigung kann der Ultraschallbonder VADB60 (Bild 1) – ein Vollautomatischer Drahtbonder – eingesetzt werden. Zwei Gerätegrundvarianten für Hybridbauelemente und Trägerstreifen erschließen ein breites Anwendungsgebiet. Als charakteristische Merkmale und Vorteile werden u. a. angegeben: drehbarer Bondkopf mit programmierbarer z-Bewegung, Aufbau von Magazin-zu-Magazin-Transportsystem oder Bauteilaufnahme, programmierbare Bondparameter und automatische Bondkontrolle. Als Draht wird AlSi mit einem Durchmesser von 25 bis 50 μm verwendet.

1


Zwei K1520 dienen als Steuerrechner.
Spitzenmodell war der vollautomatische Chipbonder VACB01 (Bild 2), der mit hoher Genauigkeit verschiedene Montagetechnologien ausführen kann. Verarbeitet werden Scheiben von 2 bis 6 Zoll Durchmesser und einer Dicke bis zu 0,6 mm. VACB01 verfügt über ein Bilderkennungs- und Kontrollsystem für automatische Chipselektion. Wichtige technologische und maschinentechnische Parameter sind frei programmierbar. Die Bondzeit liegt im Bereich von 0,1 bis 10 s. Die Chipablageunsicherheit beträgt in x,y-Richtung ± 50 bis 100 μm in Abhängigkeit von Chipabmessung und Montagetechnologie. *MP*

Technische Hochschule Magdeburg wurde Universität

Der Status einer Technischen Universität wurde am 23. März der Technischen Hochschule „Otto von Guericke" Magdeburg verliehen. Die Bildungs- und Forschungsstätte, an der bisher 16000 Direkt- und 3000 Fernstudenten ein Studium absolvierten, war 1953 als Hochschule für Schwermaschinenbau gegründet worden.
Auf der Grundlage von 15 Koordinierungs- und über 140 Leistungsverträgen arbeitet die Universität eng mit Kombinaten und Betrieben zusammen.

Neuer Schachcomputer

Der Schachcomputer CMC „Diamond" ist eine Weiterentwicklung des Schachcomputers „Chess Master". Er wurde u. a. durch den Steckplatz für eine Programmkassette zur Programmerweiterung ergänzt. Die Zugeingabe erfolgt auch hier automatisch über Sensorfelder. Der CMC hat 16 KByte ROM und 3 KByte RAM. Über eine 4stellige alphanumerische LED können die Zeit, die Anzahl der Züge sowie die Stellungsbewegung ausgegeben werden. Acht Spielstufen, zwei Analysestufen und vier Mattstufen sind einstellbar.

Informationsbank Software

Anfang 1986 wurde die *Zentrale Informationsbank Software* ZIBS gegründet, die die Mehrfachnutzung von Software-Lösungen ermöglichen soll. Gegenwärtig sind in der ZIBS rund 10000 unterschiedliche Software-Lösungen gespeichert. Jeder Betrieb ist verpflichtet, vor Beginn seiner Entwicklungsarbeiten im VEB Datenverarbeitungszentrum, Kennwort: Software, PF 408, Dresden, 8012 nach bereits vorhandenen Lösungen anzufragen bzw. Informationen über eigene Software an die Informationsbank zu übergeben.

Termine

XII. Fachtagung Mikroelektronik
Dezentrale Informationsverarbeitung bei der Automatisierung technischer Prozesse, Meßtechnik und der Bürodatenverarbeitung

WER? Ingenieurschule für Elektronik und Informationsverarbeitung
WANN? 3.–5. Mai 1988
WO? Stadthalle Görlitz
WAS? Aufzeigen von Notwendigkeiten und Möglichkeiten der dezentralen Informationsverarbeitung. Dabei stehen folgende Themenkomplexe im Vordergrund:
- Einsatz lokaler und globaler Netze in der Bürodatenverarbeitung
- lokale Netze in der Prozeßautomatisierung und in der Meßtechnik
- Software und Hardware für Mikrorechnerkopplungen
- nachnutzungsfähige Komplexlösungen unter Einsatz von Mikrorechnern für die Automatisierungstechnik
- verallgemeinerungsfähige Teillösungen aus größeren Automatisierungsprojekten
- Projekte und Einsatzerfahrungen unter Nutzung industriell gefertigter Mikrorechnersysteme
- Methoden und Erfahrungen zur Entwicklung und Implementierung von Programmsystemen für die dezentrale Informationsverarbeitung.

Die Veranstaltung ist wie immer mit einer Ausstellung gekoppelt, die gemeinsam mit dem VEB Maschinenbauhandel Berlin zu Problemen der elektronischen Meßtechnik gestaltet wird. Ferner besteht die Möglichkeit, die in den Vorträgen vorgestellten Lösungen auf dieser Ausstellung im Original vorzuführen.
WIE? Vortragsanmeldungen (mit Thema und Schwerpunkten) und Teilnahmemeldungen bis spätestens 1. September 1987 an: Ingenieurschule für Elektronik und Informationsverarbeitung „Friedrich Engels", Dr. Ulrich, Boleslaw-Bierut-Str. 1, Postfach 151, Görlitz, 8900.
Dr. Ulrich

Fertigungsorientierte Meß- und Prüftechnik

Stand und Tendenzen

Dr. Uwe Bühn
VEB Robotron-Meßelektronik
„Otto Schön" Dresden

1. Aufgaben und Ziele der Meß- und Prüftechnik

Jeder Fertigungsprozeß ist prinzipiell fehlerbehaftet, weil es

– keine fehlerfrei produzierende technologische Einrichtung
und
– keinen fehlerfrei arbeitenden Menschen
gibt.

Trotzdem besteht die Notwendigkeit, ein (nahezu) fehlerfreies Endprodukt auszuliefern und dafür eine bestimmte Zuverlässigkeit zu garantieren.
Die in den Prozeß integrierte Meß- und Prüftechnik (MuP) verfolgt deshalb zwei strategische Ziele:

- Sicherung der Erzeugnisqualität durch Erkennen und Beseitigen von Fehlern (Bekämpfung der Wirkungen) als Nahziel und
- Verbesserung der Erzeugnisqualität durch Erkennen und Beseitigen der Fehlerursachen als Fernziel.

Das für den ersten Fall gültige Trivialmodell zeigt Bild 1.
Jedes fehlerhafte Produkt zirkuliert solange in der Test- und Reparaturschleife, bis es im Rahmen der möglichen Fehlererkennung als fehlerfrei bewertet wird.
Das Modell gibt zunächst noch keine Auskunft darüber, *was, wo* und *womit* geprüft werden soll. Um diese Fragen beantworten zu können, benötigt man eine detaillierte Prüfstrategie, und diese wiederum wird ausschließlich durch die Ökonomie bestimmt.
Als Faustregel dafür ist international seit Jahren die sogenannte „Zehnerregel" bekannt (Bild 2), die besagt, daß zur Beseitigung eines in den Prozeß eingeschleppten (Bauelemente-)Fehlers von Prozeßstufe zu Prozeßstufe mit dem Faktor 10 wachsende Kosten aufzuwenden sind.
Diese Regel hat sich inzwischen als sehr grob herausgestellt, da sie die im Prozeß erzeugten Fehler und andere Fehlerquellen nicht berücksichtigt.
Eine genauere Analyse zeigt, daß es zwei Hauptfehlergruppen gibt, nämlich:

- Fehler, die als Input in den Prozeß eingesteuert und
- Fehler, die im Prozeß generiert werden.

Zur ersten Gruppe gehören neben den viel zitierten Bauelementefehlern alle Arten von Entwurfsfehlern (Schaltungsfehler, Topologiefehler, Fehler in den technologischen Vorschriften). In der zweiten Gruppe dagegen dominieren Lötfehler, Verbindungsfehler und Bestückungsfehler.
Experten haben bis zu 500 mögliche Einflußgrößen ermittelt, die von den konkreten Prozeßbedingungen abhängen. Die Optimierung des Gesamtprozesses ist daher ein aufwendiges Rechenproblem, zu dessen Lösung inzwischen ganze Softwarepakete angeboten werden (z. B. ESCAPE der Firma Orion).
International haben sich folgende Erkenntnisse durchgesetzt:

- Fehler nicht „um jeden Preis" vermeiden, aber dort erkennen und beseitigen, wo sie erzeugt werden.
- Die Prüfstrategie muß flexibel sein auf Basis einer weitgehend konfigurierbaren Hard- und Software.

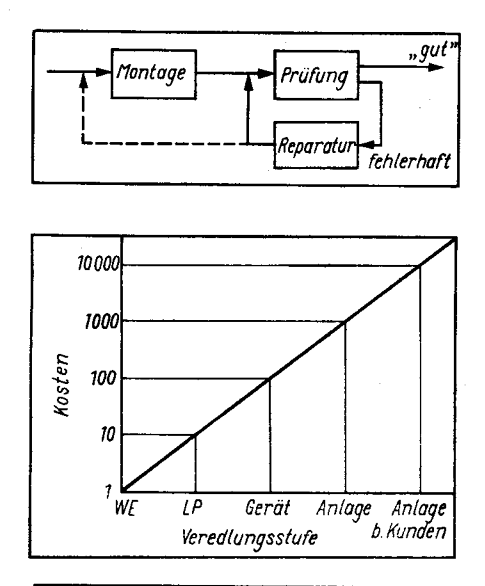
Zur Realisierung flexibler Strategien wird auf dem internationalen Markt eine Riesenpalette von Testeinrichtungen angeboten, die trotz ihrer Vielfalt auf die für die Elektronikindustrie typischen Prozeßstufen orientiert ist.
Bild 3 zeigt das klassische Prozeßmodell mit den zugeordneten Prüfprozessen und Testerkategorien.
Welche Bedeutung international der Meß- und Prüftechnik beigemessen wird, sei an drei Zahlen verdeutlicht:

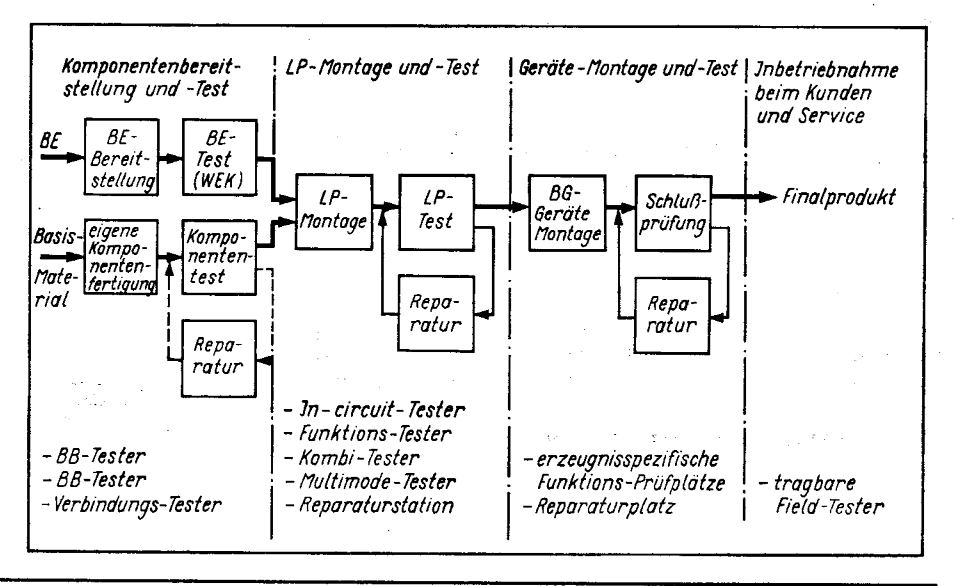
- 1982 betrug der Weltumsatz an Leiterplatten-Testern 0,5 Mrd. US-$
- 1984 erreichte er 1,3 Mrd. US-$

Bild 1 Trivialmodell eines Fertigungsabschnittes mit Montage-, Prüf- und Reparaturprozeß

Bild 2 Relative Kosten für die Lokalisierung und Beseitigung eines in den Prozeß eingeschleppten (Bauelemente-)Fehlers in Abhängigkeit vom Veredelungsgrad
WE – Wareneingang; LP – Leiterplatte montiert

Bild 3 Klassisches Modell einer Fertigungslinie für elektronische Erzeugnisse mit zugeordneten Prüfprozessen und Testeinrichtungen
BE – Bauelement
BB – Bare Board = ULP
LP – Leiterplatte





und für
- 1986 wurde ein Weltumsatz von über 3 Mrd. US-$ prognostiziert.

Der Umsatz an Bauelemente-Testern lag etwa in gleicher Größenordnung. Das sind Steigerungsraten, wie sie kein anderer Zweig der Elektronik aufweisen kann.

2. Zum internationalen Stand

Es ist nicht Anliegen dieses Beitrages, technische Daten von Testeinrichtungen verschiedener Hersteller zu vergleichen. Vielmehr sollen einige typische Erscheinungen und technisch-ökonomische Aspekte betrachtet werden.

2.1. Komponenten-Test und Wareneingangskontrolle

- *Schaltkreise*

Noch vor wenigen Jahren galt international eine 100%ige Wareneingangskontrolle (meist gekoppelt mit Burn-In) als ungeschriebenes Gesetz. Dafür werden Tester in allen Preiskategorien angeboten.
Mit dem Aufkommen der VLSI-Technik haben nun selbst finanzstarke Hersteller elektronischer Geräte nach preisgünstigeren Alternativen gesucht, da sich die erforderlichen Investitionen im Bereich jenseits von 1 Mill. US-$ bewegen.
Eine mögliche Alternative ist der Abschluß von Qualitätsvereinbarungen mit dem Bauelementehersteller, wobei letzterer gegen Aufpreis vereinbarte Qualitätsparameter garantiert und bei Nichterfüllung harte Sanktionen in Kauf nimmt.
Mit anderen Worten, beide Parteien teilen Risiko und Kosten unter dem Aspekt eines für jede Seite maximal möglichen Profits.

- *Unbestückte Leiterplatten*

Der Bare-Board-Test, d. h. die Prüfung der Rohleiterplatte auf Unterbrechungen und Kurzschlüsse, gewinnt mit zunehmendem Schwierigkeitsgrad und der verstärkt angewandten SMD-Technologie (SMD – Surface Mounted Devices, oberflächenmontierbare Bauelemente) an Bedeutung, da ein möglicher Fertigungsfehler an der fertigen Leiterplatte kaum noch korrigiert werden kann.
Spitzenerzeugnisse auf diesem Sektor verfügen über einige zehntausend Pins (Beispiel Fluke 3200 A mit 65000 Pins), wobei die maximale Pin-Anzahl heute kein elektronisches Problem mehr ist, sondern in erster Linie ein Adaptierungsproblem.
Deshalb wurde auch hier nach Lösungen gesucht, die den teuren Adapter umgehen. Dazu wurde von der Firma MANIA mit dem optischen Tester MOP 5000 eine interessante Variante auf den Markt gebracht. Dieser Tester tastet die Leiterplatte mit 8 Kameras und einer Geschwindigkeit von 1 MBit/s optisch ab und ermöglicht einen Durchsatz von 12 m^2 Fläche/Stunde. Dies entspricht etwa dem Durchsatz konventioneller Bare-Board-Tester, wobei jedoch kein spezifischer Adapter erforderlich ist. Die Sollinformation wird von einer Gutkarte abgenommen. Die Feinzentrierung der Leiterplatte erfolgt mit den Mitteln der Bildverarbeitung.

Dr.-Ing. Uwe Bühn (52) studierte von 1956–1961 an der Technischen Hochschule Ilmenau Hochfrequenztechnik. Nach einer Assistenzzeit an der gleichen Einrichtung arbeitete er von 1965 bis 1979 als Entwicklungsingenieur im damaligen VEB Funkwerk Dresden (jetzt VEB Robotron Meßelektronik, MKD). Während dieser Zeit war er im Auftrag des M für Elektrotechnik und Elektronik bzw. der UNIDO mehrmals als Spezialist in der ARÄ eingesetzt.
Seit 1980 ist er Leiter der Abteilung Meß- und Prüftechnik im VEB Robotron Meßelektronik. Dr. Bühn ist Träger zahlreicher Auszeichnungen – u. a. Verdienter Erfinder – und Inhaber von 47 Patenten.

- *Verdrahtungsrahmen*

Rückverdrahtungen, soweit sie nicht als flexible oder starre Leiterplatte ausgeführt sind, werden grundsätzlich einem Verbindungstest unterzogen. Allerdings gibt es nur wenige Firmen, die Verbindungstester anbieten (z. B. WEE/BRD). Dies läßt darauf schließen, daß die Wickelverdrahtung an Bedeutung verliert.

2.2. Board-Test

Der unwahrscheinlich harte Konkurrenzkampf, der auf dem Gebiet des Board-Tests (Prüfung bestückter Leiterplatten) von mehr als zweihundert Firmen ausgetragen wird (wobei die 4 „Großen“, nämlich GenRad, Membrain, Marconi und Zehntel über 50% des Marktes beherrschen), läßt den Schluß zu, daß die bestückte Leiterplatte (BLP) auch weiterhin die zentrale Baugruppe der Elektronik bleiben wird.
Zum besseren Verständnis der gegenwärtigen Situation seien an dieser Stelle einige Erläuterungen zu den Begriffen „Funktionstest“ und „In-Circuit-Test“ eingefügt.
Beim *Funktionstest* wird die Schaltung „am Rand“ adaptiert (meist über den Steckverbinder) und mit den notwendigen Versorgungsspannungen und Prüfsignalen beaufschlagt. Die Reaktion des Prüflings wird in der Regel ebenfalls „am Rand“ gemessen (s. Bild 4a).
Von Ausnahmen abgesehen, wird der Test beim ersten Fehler abgebrochen. Die anschließende Fehlersuche ist kompliziert und erfordert hochqualifizierte Fachkräfte.
Beim *In-Ciruit-Test* wird – wie der Name aussagt – „in der Schaltung“ gemessen. Dazu wird jeder Schaltungsknoten über ein Nadelbett adaptiert. Im Gegensatz zum Funktionstest wird die Schaltung nicht in den aktiven Zustand versetzt, sondern jedes Bauelement wird unter Ausschaltung der Einwirkung der umgebenden Schaltung isoliert für sich geprüft (siehe Bild 4b). Dazu sind raffinierte Testprinzipien entwickelt worden, die in der einschlägigen Literatur ausführlich beschrieben sind /1/. Da auch diese Prinzipien nur dann funktionieren, wenn eine 100%ige Adaptierung vorliegt und die Leiterplatte frei von Kurzschlüssen und Unterbrechungen ist, werden der Bauelementeprüfung unter Nutzung der vorhandenen Elektronik Prüfungen auf

- Kontaktsicherheit
- Kurzschlüsse und
- Unterbrechungen

vorangestellt (Trivialfehler-Test). Zu diesem Zweck wird jeder Leiterzug und jede mögliche Verbindung zwischen benachbarten Leiterzügen (Ätzfehler oder Lötfehler) als Bauelement (Widerstand) betrachtet.
Eine Sonderstellung nimmt die Prüfung integrierter digitaler Schaltkreise auf der Leiterplatte ein. Da die logische Prüfung nur bei anliegenden Versorgungsspannungen erfolgen kann, läßt sich der Einfluß der umgebenden Schaltung nur ausschalten, indem die tatsächlich anliegenden Pegel brutal überschrieben werden (als back-driving bzw. nodeforcing, d. h. Knotenüberschreibung bezeichnet). Da dieses Testprinzip den Schaltkreis thermisch stark belastet, muß die Zeit für den Test auf einige zehn bis einige hundert Mikrosekunden beschränkt werden.
Bild 5 zeigt das Grundprinzip. Hinter jedem Pin des Testers befindet sich eine aufwendige Elektronik, die program-

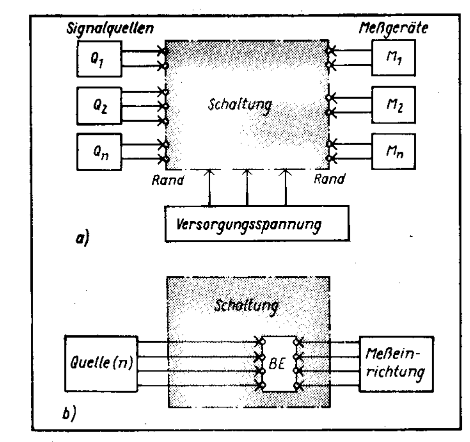


Bild 4 Prüfprinzipien
a) Prinzip der Funktionsprüfung
b) Prinzip der In-Circuit-Prüfung

mierbare Treiber- und Sensorschaltungen sowie einen Pin-Speicher (RAM) umfaßt.
Dieser Speicher enthält vor der Prüfung das Prüfmuster und nach der Prüfung das Testergebnis.
Um im Interesse einer hohen Taktfrequenz die Leitungslänge kurz halten und gleichzeitig 500 ... 2000 Pins versorgen zu können, muß die Pin-Elektronik grundsätzlich in LSI-Technik ausgeführt sein.
Die Vorteile des In-Circuit-Tests sind enorm. In einem ersten Pass (Prüfgang) werden sämtliche Trivialfehler und im nachfolgenden zweiten Pass sämtliche Bauelementefehler punktgenau ermittelt und protokolliert.
Die Programm-Erzeugung geschieht automatisch (ATG – Automatic Test Generation) und erfordert lediglich die Eingabe der Stücklisten-Daten sowie die Eingabe willkürlich festgelegter Knotennummern. Die Prüfmuster aller international eingeführten Schaltkreise sind in einer Schaltkreis-Bibliothek zusammengefaßt (einige 1000 Bibliotheksprogramme sind Stand der Technik).
Bis 1972 war der *Funktionstester* (FuT) das einzige Mittel zur BLP-Prüfung. Seine Unfähigkeit, Prozeßfehler zu lokalisieren, führte zur Erfindung des *In-Circuit-Testers* (ICT). Nachdem wenige Jahre später mit dem *Backdriving* bzw. *Node-Forcing* ein wirksames Prinzip zur Prüfung digitaler LSI-Schaltkreise auf der BLP gefunden wurde, war der Siegeszug des ICT nicht mehr aufzuhalten, und bereits 1982 wurden etwa 20 % mehr ICT als FuT verkauft.
Inzwischen bahnt sich ein Comeback des FuT an, der die Gegner des ICT frohlocken läßt.
Eine genaue Betrachtung zeigt jedoch, daß es sich hierbei um einen Funktionstest neuer Qualität handelt, wobei sich seine Schöpfer der Mittel des ICT bedienen, nämlich

– Lokalisierung und Beseitigung von Trivialfehlern mittels In-Circuit-Tests (Prescreening)
– Nutzung des Pin-Vektors
– automatische Testprogramm-Generierung (ATG) auf Basis von Schaltungsbibliotheken.

Erfolgen Prescreening und Funktionsprüfung über den gleichen (Nadelbett-) Adapter, so spricht man von einem *Kombi-Test,* dessen neueste, von Membrain kreierte Version einen simulierten Muttermaschinentest einschließt, wofür die Bezeichnung *Multimode-Test* eingeführt wurde.

• *Tester-Software*

In Testern der mittleren Preisklasse sind Software-Pakete von 10 ... 20 MByte Umfang installiert. Ein wesentlicher

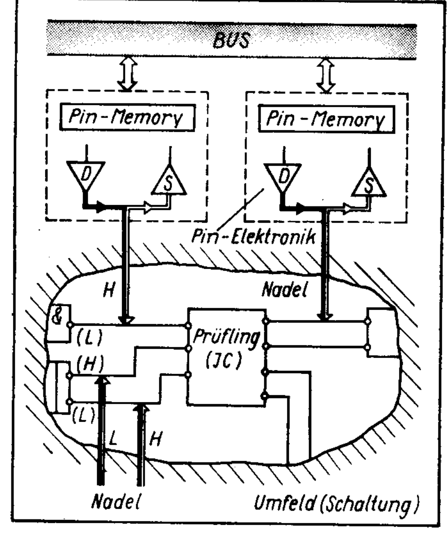


Bild 5 Prinzip des digitalen In-Circuit-Tests
D – pgb. Treiber; S – pgb. Sensor

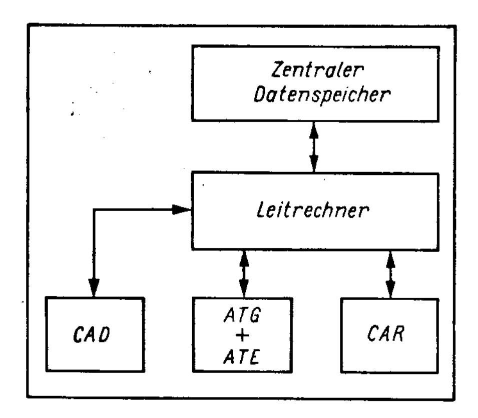


Bild 7 Modell einer zentralen Datenverwaltung
ATE – Automatic Test Equipment, automatische Test-Einrichtung
ATG – Automatic Test Generation, automatische Testprogramm-Erzeugung
CAD – Computer Aided Design, rechnergestützter Entwurf (hier: Schaltungs-, Logik- und Topologie-Entwurf)
CAR – Computer Aided Repair, rechnergestützte Reparatur

Teil davon wird von Programmen und Bibliotheken für die ATG belegt.
Während zur Zeit des klassischen Funktionstestes einige Monate für die Testprogramm-Erstellung notwendig waren, rechnet man heute mit Stunden, wobei sich die ATG stützt auf

– CAD-Daten-Eingabe zur Schaltungsanalyse
– Abnahme der Prüfmuster von einer Gutkarte (für Funktionstest).

In der Regel erfolgen Produktion und Programmerstellung im Timesharing-Betrieb. Umfangreiche Programmpakete für die Systemüberwachung, Wartung und Eigendiagnose gehören ebenfalls zum Stand der Technik.

2.3. Geräte- und Anlagen-Test

Setzt man voraus, daß alle Baugruppen und Verdrahtungsträger vor der Endmontage die geschilderten Testeinrichtungen passiert haben, so kann sich der Schlußtest auf eine Funktionskontrolle beschränken.
Dazu bedient man sich frei konfigurierbarer Prüfplätze, die aus Geräten mit Standard-Interface (z. B. IEC 625) und einem intelligenten Controller bestehen.
Da jedoch moderne elektronische Geräte zunehmend über eigene Intelligenz verfügen, wird die Schlußprüfung immer mehr durch das Erzeugnis selbst durchgeführt, indem entsprechende Inbetriebnahme-, Wartungs- und Diagnose-Routinen transient oder resident implementiert werden.

3. Zum internationalen Trend

Betrachtet man Tester und andere Prüfeinrichtungen losgelöst von ihrer Einsatzumgebung, so lassen sich natürlich technische *Trendmerkmale* extrahieren, die mehr oder weniger auch für andere Gebiete der Elektronik gelten, wie etwa

• digitale Signalerzeugung und -analyse
• dezentrale Intelligenz mit weitgehender Vorverarbeitung der anfallenden Daten (Daten-Kompression)
• Einsatz von VLSI und SMD
• Konzentration auf die Programmiersprachen BASIC, ATLAS und PASCAL bei der Software für FuT
• ATG für ICT mit Schaltkreis-Programm-Bibliotheken und für FuT mit Schaltungs-Programm-Bibliotheken.

Die eigentliche Revolution vollzieht sich jedoch gegenwärtig im Überbau, d. h. in der Ebene des übergeordneten Informationsaustausches.
Dafür stehen international die Schlagworte „Factory of the Future“ (Fabrik der Zukunft) und CADMAT (Computer Aided Design, Manufacture and Automatic Test).
Dabei sind mehrere Erscheinungen erkennbar, die letztlich auf das eingangs erwähnte Ziel „Beseitigung der Fehlerursachen“ gerichtet sind.

1. Aspekt

Beim Entwurf der ATG für den Prüfprozeß und beim Reparaturprozeß erfolgt ein Zugriff auf Daten gleichen Inhalts (siehe Bild 6 – Titelbild).
Diese Daten sind gegenwärtig meist auf unterschiedlichen Datenträgern an unterschiedlichen Orten untergebracht. Jede Änderung bedingt viele Kopien und damit hohe Kosten.
Es liegt daher nahe, den Zwischenspeicher Papier bzw. die noch übliche Off-Line-Kopplung durch einen zentralen elektronischen Speicher (Festplatte) zu ersetzen, mit folgenden Merkmalen:
• zentrale Datenverwaltung
• jeder (Teilprozeß) hat Zugriff

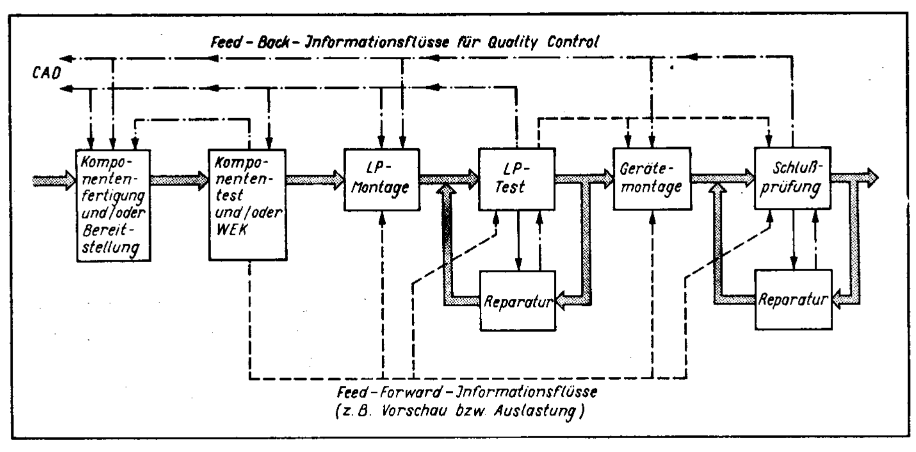


Bild 8 Material- und Informationsflüsse in und über einer Fertigungslinie

- jeder (Teilprozeß) teilt aktuelle Erkenntnisse der Zentrale mit, und
- Änderungen erfolgen nur einmal am zentralen Ort (s. Bild 7).

Für das Prüfobjekt (z. B. BLP) bedeutet das:

- Jedes Teilprodukt erhält eine elektronische Karteikarte mit seinem Lebenslauf, der in der Zentrale verwaltet wird.
- ATE (Automatische Test-Einrichtung) und CAR (Computergestützte Reparatur) aktualisieren diesen Lebenslauf bis zum „Tod" des Produkts.

Voraussetzung dafür ist die maschinelle Identifikation des Prüflings hinsichtlich Typ und laufender Nummer (z. B. mittels Balkenkode).

2. Aspekt

Prüf- und Reparaturprozeß liefern eine Fülle von Daten an die Zentrale (Bild 7), die sich bei entsprechender statistischer Auswertung nutzen lassen für
– die Qualitätssteuerung (bzw. -regelung) der Fertigung, indem gehäuft auftretende Fehler über einen Prozeß-Manager dem Verursacher signalisiert werden (Quality Control)
und
– eine Entwurfskorrektur, nämlich dann, wenn immer wieder auftretende dynamische Fehler beim FuT auf einen Entwurfsfehler schließen lassen.
Im zuletzt genannten Fall laufen in dem On-Line-gekoppelten System (Bild 7) folgende Teilprozesse ab:

– Korrektur der Schaltung
– Korrektur der Topologie
– Korrektur der Stückliste
– Korrektur der Materialbestellung
– Auslösen eines neuen ATG-Laufes

zwecks Aktualisierung des Testprogramms.
Es versteht sich von selbst, daß Quality Control nur dann funktioniert, wenn zwischen Montage- und Prüfprozeß keine Totzeiten liegen, wenn also der Prüfprozeß der dem Montageprozeß unmittelbar folgende Arbeitsgang einer bestimmten Prozeßstufe ist.
Auch Design Control ist nur dann sinnvoll, wenn eingehende Materialströme mit geringen Totzeiten beeinflußbar sind.
Bild 8 zeigt die Material- und Informationsflüsse einer kompletten Fertigungslinie. Neben den Feed-back-Datenflüssen für Quality- und Design-Control werden auch Informationen an nachgelagerte Fertigungsabschnitte geliefert (Feed-forward), zum Beispiel, um die zu erwartenden Typen oder Losgrößen anzukündigen.

3. Aspekt

Ein in einer weiteren Informationsebene angeordneter Leitrechner, der die Informationen mehrerer Fertigungslinien auswertet, kann – obwohl er eigentlich für CAM-Funktionen gedacht ist – zur Erkennung von Fehlern genutzt werden, die wegen zu geringer Häufigkeit in einer einzigen Prozeßlinie noch keine Warnung auslösen würden, wie zum Beispiel eine fehlerhafte Bauelemente-Charge.
Natürlich besteht jedoch seine Hauptaufgabe darin, Informationen zu sammeln und Aktivitäten auszulösen hinsichtlich
– Auslastung bestimmter Fertigungsabschnitte
– Qualität bestimmter Fertigungsabschnitte
– automatischer Material-Nachbestellung oder Stornierung
– Anforderung Tester-Service
– operativer Steuerung.

4. Aspekt

Ein weiterer Aspekt, der sich etwa seit 1984 zeigt, besteht in der Nutzung der Ergebnisse auf dem Gebiet der künstlichen Intelligenz (speziell Expertensysteme).
Dabei werden aufgetretene Fehlerbilder und erfolgreiche Reparaturaktivitäten in eine Bibliothek eingetragen. Mit dem Wachsen dieser Bibliothek können der Reparaturstation in zunehmendem Maße Erfolgsstrategien mitgeteilt werden.
Außer für die Reparatur des Teilprodukts wird diese Strategie auch für Fehler am Tester selbst genutzt. Dies kompensiert den leichten Abfall der Dauerverfügbarkeit mit zunehmendem Lebensalter.

- *Das Adaptierungsproblem*

Die ständig steigende Packungsdichte, der Übergang zu SMD-bestückten Leiterplatten und die Verkleinerung der Rastermaße (≤ 1,25 mm) lassen die Adaptierung der Leiterplatte zu einem Problem werden, das mit den klassischen Nadeladaptern in Zukunft nicht mehr lösbar sein wird und weltweit zu neuen Überlegungen zwingt. Dabei wird neben der fieberhaften Suche nach neuen Adaptierungsprinzipien zunehmender Wert auf eine prüfgerechte Gestaltung der Leiterplatte gelegt (Minimierung der Testpunktanzahl, Schaffung abgesetzter Adaptierungsflächen usw.). Der Umschlag in eine neue Qualität deutet sich an, aber eine elegante Lösung steht im Moment noch aus.

Literatur

/1/ Recherche D 137 B4: 42 55.8432. Informations-Leitstelle des VEB Kombinat Robotron

✉ KONTAKT ☎

VEB Robotron Meßelektronik „Otto Schön"
Dresden, PSF 211, Dresden, 8012;
Tel. 4875624

Chip-Prüfung mit Ultraschall-Mikroskop

Mit Ultraschall arbeitet ein neues Mikroskop des japanischen Unternehmens Olympus Optical Co. und erreicht damit eine Auflösung von 0,35 Mikrometern. Als Einkopplungsmedium wird nicht Wasser, sondern Flüssigstickstoff verwendet. Dabei werde die Schallausbreitungsgeschwindigkeit abgebremst und zugleich die Schallfrequenz auf eine Milliarde Hertz angehoben (üblich ist die Arbeit mit 400 Millionen Hertz). Die Bilder werden, wie die Zeitschrift „New Technology Japan" berichtet, aus den Reflexionen der Ultraschallwellen zusammengesetzt. Um Meßfehler weitgehend auszuschließen, wird bei diesem Mikroskop nicht die zu untersuchende Probe bewegt, sondern das Objektiv. Zusätzlich werden die Bilder durch computergestützte Bildkontrolle verbessert. Die japanische Firma sieht Einsatzmöglichkeiten für dieses Mikroskop unter anderem in der Kontrolle von Chips. Einerseits können damit die elektrischen Anschlüsse der Chips untersucht werden, andererseits lassen sich auch kleinste Aluminiumtröpfchen von 0,5 Mikrometer Durchmesser aufspüren, die mit Elektronenmikroskopen nicht zu entdecken sind. Außerdem besteht bei der Arbeit mit Elektronenstrahlen die Gefahr, daß Chip-Strukturen beschädigt werden.

Software für Mehrmikrorechner-systeme

Bernd Klühe
Akademie der Wissenschaften der DDR,
Zentralinstitut für Kybernetik
und Informationsprozesse,
Institutsteil Dresden

1. Rechnerverbundsysteme in der Automatisierungstechnik

Mit der Nutzung der Mikrorechentechnik für komplexe Aufgaben ergibt sich, im Gegensatz zum Mikrorechnereinsatz für Einzelaufgaben, die Notwendigkeit des Einsatzes von Rechnerverbundsystemen. Folgende Merkmale sind dafür wesentlich /1/, /2/:

- Erhöhung der Leistung durch Parallelarbeit mehrerer Rechner (Lastverbund)
- geteilte Nutzung von Ressourcen (Ressourcenverbund)
- dezentraler, aufgabenbezogener Mikrorechnereinsatz
- aufgabenbezogene Funktionsteilung zur modularen Strukturierung des Gesamtsystems (Flexibilität, Änderbarkeit, Senkung der Softwarekosten)
- Erhöhung der Fehlertoleranz (Verfügbarkeitsverbund).

Die Gesamtleistungsfähigkeit gekoppelter Systeme wird entscheidend durch die Leistungsfähigkeit des Kommunikationssystems beeinflußt, siehe u. a. /2/ und /3/. Im folgenden werden wesentliche Probleme der Betriebssystemgestaltung busgekoppelter Mehrmikrorechnersysteme (MMRS), als eine spezielle Problemklasse der Rechnerverbundsysteme, untersucht und der Gestaltung einer Mehrmikrorechner- Kommunikationssoftware (MKS) zugrunde gelegt.

2. Gestaltung wesentlicher Betriebssystemkomponenten

2.1. Hardwareumgebung

Für die in den folgenden Kapiteln untersuchten Probleme der Betriebssystemgestaltung für MMRS gelten folgende Hardwarevoraussetzungen:

- MMRS mit paralleler Buskopplung
- Jeder Rechner des MMRS verfügt über lokale Ressourcen (z. B. Speichermoduln, externe Geräte).
- Es existiert mindestens paarweise zwischen 2 Rechnern des MMRS gemeinsam erreichbarer RAM-Speicher (Dual-Port-RAM oder Global-RAM).
- Es existieren Möglichkeiten der Signalübermittlung zwischen verschiedenen Rechnern (Interruptübermittlung).

Die praktischen Untersuchungen wurden mit Hardwaremoduln des MMS-16 durchgeführt. Das im MMS-16 definierte Hardwaresystem erfüllt die oben genannten Bedingungen und hat außerdem einige Eigenschaften, die für die Rechnerkopplung vorteilhaft sind (z. B. Trennung von Systembus und Residentbus). In Bild 1 wird die betrachtete Hardwarestruktur dargestellt.

2.2. Aufgabenstellung

Die Kommunikation in einem MMRS stellt eine Verallgemeinerung der Kommunikation im Einrechnersystem dar. Es ergibt sich unter den genannten Hardwarevoraussetzungen folgende Aufgabenstellung:

- Realisierung des Datentransfers zwischen Tasks, die auf unterschiedlichen Rechnern arbeiten.
- Die Tasks jedes Rechners arbeiten unter Steuerung eines lokalen Echtzeitbetriebssystems.
- Die Kommunikation im MMRS muß den Echtzeitbedingungen des Gesamtsystems gerecht werden.
- Schaffung der Voraussetzungen für die Kommunikation in heterogenen MMRS
  · bezüglich Hardware
  (z. B. Einsatz unterschiedlicher Prozessoren)
  · bezüglich Software
  (z. B. Einsatz unterschiedlicher lokaler Betriebssysteme)
- Konfigurierbarkeit
- Generierbarkeit.

Aus dieser Aufgabenstellung ergibt sich die in Bild 2 am Beispiel eines MMRS mit 3 Rechnern dargestellte Softwarestruktur.
Im Gegensatz zur Kommunikation innerhalb eines Rechners, u. a. /4/ ... /8/, kann im MMRS mit lokaler Taskverwaltung eine direkte Verbindung zwischen sendender Task und empfangender Task nicht hergestellt werden. Deshalb müssen vom Kommunikationssystem logische Kanäle zwischen den Rechnern verwaltet werden, über welche die Kommunikationsanforderungen übertragen werden (siehe 2.3.1). Wie im Einrechnersystem ist im MMRS im allgemeinen eine Entkopplung der Sender und Empfänger erforderlich, um eine möglichst weitgehende Parallelarbeit von Sendern und Empfängern zu ermöglichen. Die Übertragungsanforderungen zwischen den Rechnern werden in den logischen Kanälen mit Hilfe von Warteschlangen verwaltet (siehe 2.3.2. und 2.3.3.).
Die Verbindung zur Empfängertask wird über ein Port hergestellt. Dabei ist ein Port eine definierte Schnittstelle, mit der das Ende eines Kommunikationskanals zwischen Prozessen (Tasks) bezeichnet wird. Ein Port dient dem Einrichten und Identifizieren eines Kommunikationskanals /2/.
Wenn in einem MMRS lokale Speicherbereiche verfügbar sind, so sind diese weitestgehend für die Datenspeicherung zu nutzen, um die Belastung des Systembusses zu minimieren. Daraus ergibt sich als weiterer Unterschied zur Kommunikation im Einrechnersystem, daß im MMRS mit lokalem BS im allgemeinen die Nachrichten physisch kopiert werden müssen, um sie im lokalen Speicher des Zielrechners verarbeiten zu können (siehe 2.3.4.). Im Gegensatz dazu ist im Einrechnersystem nur die Übermittlung der Nachrichtenadresse und der Nachrichtenlänge zur Zieltask erforderlich.

2.3. Wesentliche Komponenten des Mehrrechnerbetriebssystems

2.3.1. Lokale Funktionen

Die lokalen Funktionen des Kommunikationssystems dienen der Übermittlung der Übertragungsanforderungen der Anwendertasks an die Kanäle zwischen Sende- und Zielrechner sowie dem Empfang von Nachrichten von anderen Rechnern mit anschließender Weiterleitung zu den Empfängertasks. Diese Funktionen werden unter Nutzung der Dienste der Warteschlangenverwaltung (siehe 2.3.2.) und des Signalsystems (siehe 2.3.3.) realisiert.
Weitere lokale Funktionen sind die Zwischenspeicherung von Übertragungsanforderungen bei zeitweise überlasteten Übertragungskanälen und die Steuerung der Pufferverwaltung (siehe 2.3.4.).
Die Realisierung der lokalen Funktionen erfolgt unter Steuerung des lokalen Echtzeitbetriebssystems.

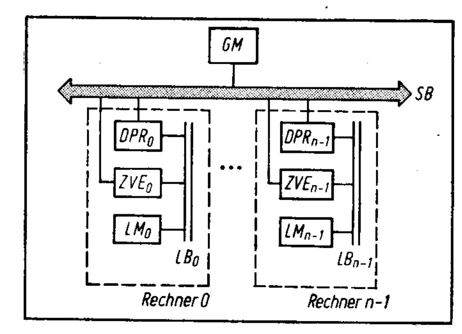


Bild 1 Hardwarestruktur
n – systembusgekoppelte Mikrorechner
DPR – Dual-Port-RAM
GM – Globalspeicher
LB – Lokalbus
LM – Lokalspeicher
SB – Systembus
ZVE – Prozessor

2.3.2. Warteschlangenverwaltung

Als Warteschlangen (WS) werden Datenstrukturen bezeichnet, die eine Anzahl von Anforderungen einer bestimmten Klasse speichern können. Sie dienen als Puffer zwischen Produzenten und Konsumenten dieser Anforderungen.
In modernen Echtzeitbetriebssystemen werden WS für einen bestimmten Zweck als Objekttypen aufgefaßt (z. B. bei BOS–1810: WS zur Verwaltung von Kommunikationsanforderungen haben den Objekttyp MAILBOX, WS zur Verwaltung des exklusiven Zugriffs zu kritischen Bereichen haben den Objekttyp REGION).
Im vorliegenden Fall der Rechnerkopplung dienen die WS der Speicherung von Kommunikationsanforderungen zwischen Rechnern. In Einrechnerbetriebssystemen erfolgt die Verwaltung der WS durch Routinen des Betriebssystemkerns, dadurch kann der exklusive Zugriff zu den Datenstrukturen der WS bei jeder Veränderung durch die innere Organisation des lokalen Betriebssystems gesichert werden. Im MMRS mit lokalen Betriebssystemen auf jedem Rechner ist die Steuerung der Zugriffsrechte zu den WS zwischen den Rechnern keine lokale Funktion mehr.
Durch den exklusiven Zugriff zu WS zwischen verschiedenen Rechnern treten im Falle des Zugriffskonfliktes Wartezeiten bei den Rechnern auf, die auf das Zugriffsrecht zu einer WS warten müssen, weil das Zugriffsrecht bereits einem anderen Rechner übergeben wurde.
Für die logische Anordnung von WS zwischen Rechnern werden die in Bild 3 dargestellten prinzipiellen Varianten betrachtet. Im Fall a) existiert im MMRS genau eine WS für alle Kommunikationsanforderungen zwischen den Rechnern. Bei b) hat jeder Rechner eine Empfangswarteschlange, in die alle anderen Rechner ihre Kommunikationsanforderungen eintragen. Im Fall c) gibt es zwischen jeweils 2 Rechnern ein Paar von WS, die jeweils für eine Kommunikationsrichtung benutzt werden.
Aus Bild 3 ist ersichtlich, daß die Wahrscheinlichkeit des Zugriffskonfliktes zu WS zwischen verschiedenen Rechnern von der strukturellen Anordnung der WS abhängig ist. Dabei zeigt sich, daß die Wahrscheinlichkeit des Auftretens von Zugriffskonflikten in der Struktur c) am geringsten ist. Deshalb wurde diese Struktur gewählt, um die möglichen Wartezeiten durch Zugriffskonflikte zu minimieren. In dieser Struktur existiert für jede WS genau eine schreibende Task und eine lesende Task auf zwei verschiedenen Rechnern. Für diese spezielle Art einer WS konnte, in Anlehnung an das MULTIBUS INTERPROZESSOR PROTOCOL (MIP) /9/, ein Algorithmus gefunden werden, bei dem die Sicherung des exklusiven Zugriffs beim Bearbeiten der WS durch Semaphorvariablen nicht notwendig ist. Die dabei aufgetretenen Probleme des Softwareentwurfs wurden durch Nutzung der Methoden der Theorie der Petri-Netze gelöst /10/. Es wurden Petri-Netze mit problembezogener Bewertung zur Verhaltensbeschreibung der Softwarekomponenten der Warteschlangenverwaltung genutzt.
Durch das hohe Maß an Anschaulichkeit, das diese Darstellung im Vergleich zu üblichen Methoden auch bei Parallelprogrammierung bietet, wird der Entwurf erheblich unterstützt und die Programmdokumentation erleichtert.

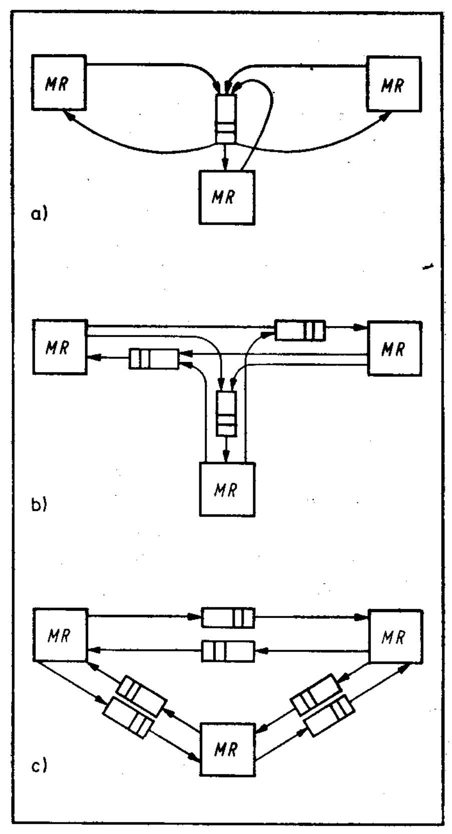


Bild 3 Warteschlangenstruktur in Mehrmikrorechnersystemen

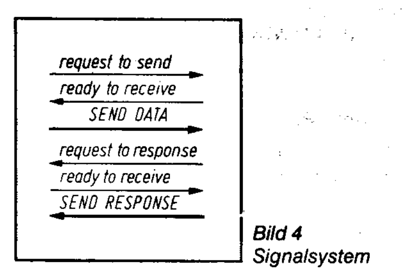


Bild 4 Signalsystem

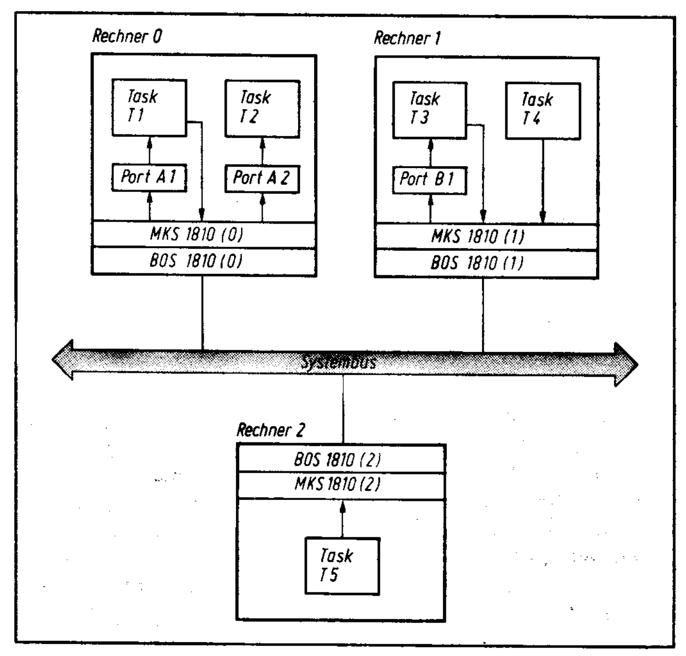


Bild 2 Beispiel eines Systementwurfs mit Task- und Portverteilung

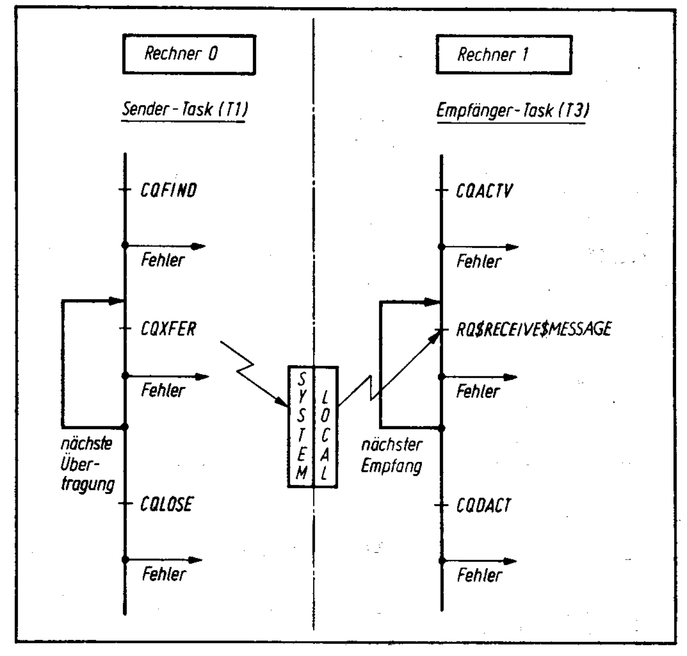


Bild 5 Ablauf der Taskkommunikation

2.3.3. Signalsystem

Mit der Verwaltung von WS zwischen den Rechnern ist die Funktion eines Signalsystems verbunden. Vom Einrechnerbetriebssystem wird eine auf eine Nachricht wartende Task sofort bei Eintreffen einer Nachricht in den READY-Zustand versetzt. Um im MMRS das uneffektive zeitzyklische Testen aller WS zu verhindern, ist nach Eintragung eines Sendeauftrages in eine WS der Zielrechner über diese Veränderung zu informieren, damit der Auftrag aus der WS entnommen und zur Zieltask weitergeleitet werden kann.
Die Signalisierung zwischen den Rechnern erfolgt über Interrupt. Es ist die Anzahl der für einen Übertragungsvorgang erforderlichen Interrupts zu minimieren.
In Bild 4 wird die allgemeine Form eines Handshake-Protokolls für den Signalfluß dargestellt. Bei geeigneter Organisation der WS sind nur die Signale 'SEND DATA' und 'SEND RESPONSE' notwendig. Bei aufeinanderfolgender Übertragung mehrerer Datenblöcke sind diese Signale nur zur Einleitung eines Übertragungszyklus erforderlich.
Die pysikalische Realisierung der Interruptübermittlung ist hardwareabhängig.

2.3.4. Pufferverwaltung

Wie in 2.2. erläutert wurde, müssen im MMRS mit lokalem BS im allgemeinen die zu übertragenden Nachrichten unter Benutzung gemeinsam erreichbarer Speicherbereiche umkopiert werden. Soll die parallele Übertragung verschiedener Nachrichten zwischen 2 Rechnern möglich sein, müssen in den gemeinsam erreichbaren Speicherbereichen verschiedene Nachrichtenblöcke verwaltet werden. Diese Aufgabe wird durch eine dynamische Pufferverwaltung innerhalb der gemeinsamen Speicherbereiche erfüllt.

2.4. Implementierung der MKS

2.4.1. Allgemeine Eigenschaften

Die Implementierung der Kommunikationssoftware MKS mit den in 2.3. beschriebenen Eigenschaften erfolgte auf der Basis der Hardware des Systems MMS-16 als Erweiterung der lokalen Echtzeitbetriebssysteme BOS-1810 (kompatibel zu iRMX-86) und MRTS-700 /11/.
Die Implementierung erfolgte in der Programmiersprache PL/M-86. Die entstandenen Softwaresysteme MKS-1810 und MKS-700 sind während der Systemgenerierung für jeden Rechner des MMRS den konkreten Anforderungen anzupassen. Dazu ist in Datenstrukturen die Konfiguration des MMRS zu beschreiben. Folgende wesentliche Parameter werden zur Beschreibung des MMRS benutzt /12/:
– Angabe konkreter Speicherbereiche für die Informationsübermittlung zwischen den Rechnern
– Zuordnung logischer Portadressen zu den Rechnern
– Spezifizierung der realisierten Interruptstrukturen.

Dipl.-Math. Bernd Klühe (34) studierte von 1970 bis 1974 Mathematik an der TH Magdeburg. Er diplomierte 1974 mit einer Arbeit zu Lösungsmethoden spezieller ganzzahliger Optimierungsprobleme.
Von 1974 bis 1983 arbeitete er im VEB Mikromat Dresden an Problemen der Rationalisierung der Arbeit des betrieblichen Rechenzentrums.
1983 nahm er die Tätigkeit im ZKI der AdW auf und beschäftigt sich gegenwärtig mit speziellen Softwareaufgaben auf dem Gebiet der Mehrmikrorechner-Betriebssysteme.

Dem Anwender stehen Dienste zum Eröffnen und Schließen der logischen Verbindungskanäle und zum Datentransfer zur Verfügung.

2.4.2. Programmfunktionen

Die durch MKS-1810 implementierten Systemrufe ermöglichen den Tasks, Verbindungen zur Nachrichtenübertragung zu eröffnen und Ports zu aktivieren, Nachrichten (Einzelnachricht max. 64 KByte lang) zu senden bzw. zu empfangen und die Ports nach beendeter Übertragung wieder zu deaktivieren.
Eine sendende Task benutzt folgende MKS-Rufe:

CQFIND
Es wird eine Verbindung (connection) zu einem zur Generierungszeit definierten Kommunikationsport eröffnet.

CQXFER
Es wird eine Nachrichtenübertragung an ein zuvor durch CQFIND gebundenes Port eingeleitet, wobei die Nachricht zuvor in einem (lokalen oder gemeinsamen) Pufferspeicherbereich aufgebaut wurde.

CQLOSE
Eine durch CQFIND eröffnete Verbindung wird wieder freigegeben.

Eine empfangende Task verwendet folgende Rufe:

CQACTV
Es wird eine Verbindung zum Nachrichtenempfang an einem Systemport eröffnet, indem u. a. eine (lokale) Mailbox eingerichtet wird.

RQ$RECEIVE$MESSAGE
Dient dem Nachrichtenempfang an der zuvor durch CQACTV eingerichteten Mailbox.

CQDACT
Die durch CQACTV eröffnete Verbindung wird wieder freigegeben.

Dabei sind CQACTV und CQDACT MKS-Rufe, während RQ$RECEIV E$MESSAGE ein Ruf des Echtzeitbetriebssystems BOS 1810 ist.
Eine Nachrichtenübertragung besteht nun senderseitig darin, daß mit den Rufen CQFIND, CQXFER und CQLOSE ein Kommunikationsport gebunden wird, ein- oder mehrfach Nachrichten zu ihm gesendet werden (einschließlich Fehlercodeauswertung und Zeitüberwachung) und danach die Verbindung wieder aufgelöst wird.
Analog werden empfängerseitig in einer Abfolge der Rufe CQACTV, RQ$RECEIV E$MESSAGE und CQDACT das Port aktiviert, die Nachricht(en) empfangen und das Port wieder deaktiviert.
Eine Task kann gleichzeitig Sender und Empfänger sein (im Spezialfall auch mit dem gleichen Taskpartner).
In Bild 5 wird der prinzipielle Ablauf einer Taskkommunikation mittels MKS dargestellt.

Literatur

/1/ Fathi, E. T.; Krieger, M.: Multiple Mikroprocessor Systems. Computer No. 3, 1983
/2/ Löffler, H.: Rechnerverbundsysteme. Akademie-Verlag, Berlin, 1984
/3/ Werner, D.: Aufbau eines verteilten Echtzeitbetriebssystems KOMINET für Mikrorechnerverbundsysteme. messen-steuern-regeln 27 (1984) 8
/4/ Ben-Ari, M.: Prinziples of Concurrent Programming. Prentice-Hall International, 1982
/5/ Hansen, P. B.: Konstruktion von Mehrprozeßprogrammen. R. Oldenbourg Verlag, München/Wien, 1981
/6/ Pieper, F.: Einführung in die Programmierung paralleler Prozesse. R. Oldenbourg Verlag, München/Wien, 1977
/7/ Levi, P.: Betriebssysteme für Realzeitanwendungen. Datakontext-Verlag, Köln, 1981
/8/ Werner, D.: Programmierung von Mikrorechnern. VEB Verlag Technik, Berlin 1983
/9/ Multibus Message Exchange Referenz Manual. Firmenschrift Intel, Order No. 144912, 1982
/10/ Arendt, F.; Klühe, B.: Verhaltensbeschreibung von Softwarekomponenten mit Hilfe von Petri-Netzen am Beispiel der Warteschlangenverwaltung in einem Mehrrechner-Betriebssystem. Studie, ZKI der AdW, IT Dresden, 1986
/11/ Echtzeitbetriebssystem MRTS-700. Systemdokumentation. VEB Numerik „Karl Marx", Karl-Marx-Stadt, 1986
/12/ Despang, G., u. a.: Architektur und Anwendung eines modularen echtzeitfähigen Betriebssystems für ein 16-Bit-Mehrmikrorechnersystem. 14. Arbeitstagung Entwurf von Schaltsystemen mit Systementwurf ZKI der AdW, IT Dresden, 1985

✉ KONTAKT ☎
Akademie der Wissenschaften der DDR, Zentralinstitut für Kybernetik und Informationsprozesse, Institutsteil Dresden, Haeckelstraße 20, Dresden, 8027; Tel. 4633122

Schneller 12-Bit-A/D-Wandler C 574 C

Klaus Christen
VEB Halbleiterwerk Frankfurt (Oder)

Der C 574 C ist ein vollständiger Sukzessiv-Approximations-A/D-Wandler mit Mikroprozessor-Interface, einer Auflösung von 12 Bit und einer Umsetzzeit von max. 40 µs. Er kann ohne zusätzliche Treiber- oder Peripherieschaltkreise mit 8- oder 16-Bit-Mikroprozessoren zusammengeschaltet werden. Dabei werden Lese- und Umsetzstartkommandos unmittelbar dem Steuerbus entnommen. Die Ausgangsdaten können als ein 12-Bit-Wort oder als zwei 8-Bit-Byte gelesen werden. Der Wandler verarbeitet unipolare Eingangsspannungen von 0 bis 10 V bzw. 0 bis 20 V oder bipolare Eingangsspannungen von −5 V bis +5 V bzw. −10 V bis +10 V.

Blockschaltbild

Bild 1 zeigt das Blockschaltbild des C 574 C. Für den Analog-Digital-Wandler wurde eine leistungsfähige und kostengünstige Zwei-Chip-Lösung gewählt. Auf einem Chip befinden sich der 12-Bit-D/A-Wandler, eine vergrabene Zenerdioden-Referenzspannungsquelle mit niedrigem Temperaturkoeffizienten, die Eingangsspannungsbereichswiderstände und der Bipolar-Offset-Widerstand.
Das zweite Chip beinhaltet das I²L-Sukzessiv-Approximationsregister, die Steuerlogik des Umsetzers, den Taktgenerator, die Tristate-Ausgangstreiber und den Komparator.

Steuersignale

Die Steuersignale CE, $\overline{CS}$, R/$\overline{C}$, A0 und 12/$\overline{8}$ steuern die Lese- und Umsetzoperationen des Wandlers.
Die Auslösung einer Operation ist mit einer Flanke eines der Steuersignale CE, $\overline{CS}$ oder R/$\overline{C}$ möglich.
In Abhängigkeit vom R/$\overline{C}$-Signal wird eine Datenleseoperation (R/$\overline{C}$ = H) oder eine Analogwertumsetzung (R/$\overline{C}$ = L) ausgeführt. Die Steuersignale A0 und 12/$\overline{8}$ steuern die Länge des Umsetzzyklus und das Datenformat.
Der A0-Eingang wird vorzugsweise mit dem niederwertigsten Bit des Adreßbusses verbunden. H-Potential an A0 während des Umsetzstarts bewirkt die Ausführung eines vollständigen 12-Bit-Umsetzzyklus, L-Potential an A0 dagegen die Ausführung eines verkürzten 8-Bit-Umsetzzyklus. Während der Datenleseoperation bestimmt A0, ob durch die Tristate-Ausgangstreiber die 8 MSB des Umsetzergebnisses (A0 = L) oder die 4 LSB (A0 = H) freigegeben werden. Das 12/8-Signal bestimmt, ob die Ausgangsdaten als 12-Bit-Wort (12/$\overline{8}$ verbunden mit U_{CC3}) oder als zwei 8-Bit-Worte (12/$\overline{8}$ verbunden mit Digitalmasse) ausgegeben werden.

In 8-Bit-Bus-Applikationen (12/$\overline{8}$ an Digitalmasse) sind die Bit-Ausgänge der 4 LSB (Pin 16 bis Pin 19) mit den Bit-Ausgängen der 4 MSB (Pin 24 bis Pin 27) zu verbinden. Es wird dann im Format gemäß Tafel 1 ausgegeben.

Ist der Umsetzvorgang einmal ausgelöst, werden weitere Umsetzstartkommandos bis zur Beendigung des Umsetzvorganges ignoriert. Die Ausgangstreiber können während der Umsetzung nicht aktiviert werden. Sie sind – wie auch im inaktiven Zustand des Wandlers – hochohmig. Die Wirkung der Steuersignale ist zusammenfassend in Tafel 2 dargestellt.

Kenndaten

Ausgewählte Kenngrößen des C 574 C sind in Tafel 3 dargestellt. Die Kenngrößen gelten für den gesamten Betriebstemperaturbereich ($\vartheta_a = 0 \ldots 70°C$) und berücksichtigen daher den Temperaturkoeffizienten der Fehlerkenngrößen. Die einzuhaltenden Betriebsbedingungen sind in Tafel 4 angegeben.

Applikation

- Die Betriebsspannungen sind nahe am Schaltkreis mit 47 µF und 100 nF (keramischer Scheibenkondensator) abzublocken.
- Die Betriebsspannung U_{CC3} (+5 V) ist nach Digitalmasse (Pin 15), die Betriebsspannungen U_{CC1}, U_{CC2} (+15 V, −15 V) sind nach Analogmasse abzublocken. Die Betriebsspannungen müssen gut stabilisiert und frei von hochfrequenten Störungen sein.

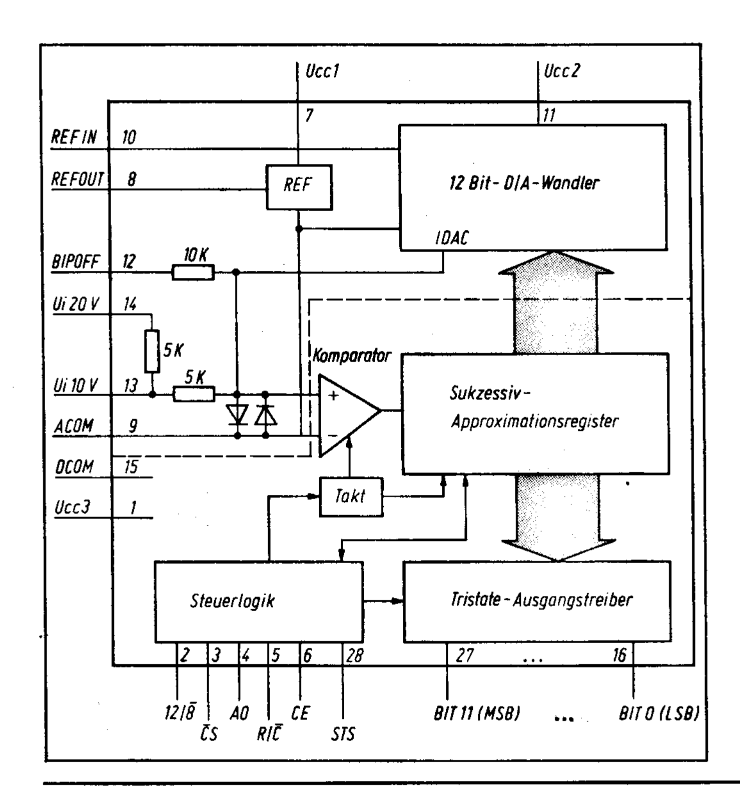
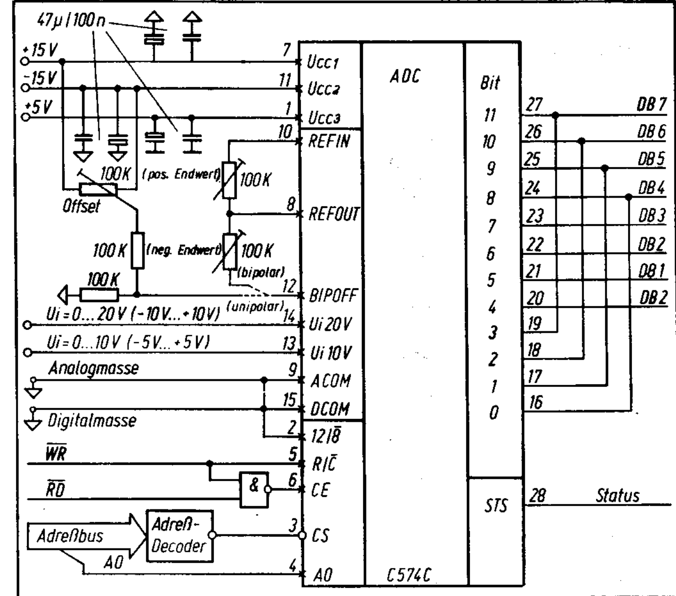


Bild 1 Blockschaltbild

Bild 2 Zusammenschaltung des C 574 C mit dem U 880 D



	D7	D6	D5	D4	D3	D2	D1	D0
(Pin	27	26	25	24	23	22	21	20)
xxx0 (gerade Adr., A0 = L)	B11 (MSB)	B10	B9	B8	B7	B6	B5	B4
xxx1 (ungerade Adr., A0 = H)	B3	B2	B1	B0 (LBS)	L	L	L	L

Tafel 1 Ausgabeformate ▲

Tafel 2 Wirkung der Steuersignale ▼

CE	$\overline{CS}$	R/$\overline{C}$	12/$\overline{8}$	A0	Wirkung
L	X	X	X	X	inaktiv
X	H	X	X	X	inaktiv
H	L	L	X	L	Auslösung einer 12-Bit-Umsetzung
H	L	L	X	H	Auslösung einer 8-Bit-Umsetzung
H	L	H	verbunden mit Pin 1	X	Freigabe der Datenausgänge 12 Bit parallel
H	L	H	verbunden mit Pin 15	L	Freigabe der 8 MSB
H	L	H	verbunden mit Pin 15	H	Freigabe der 4 LSB und Belegung der 4 folgenden Bit mit L-Signal

Kenngröße	Kurzzeichen	Einheit	Einstellwerte	Kleinstwerte	Größtwerte
Stromaufnahme	I_{CC3}		$U_{CC3} = 5{,}5\,V; U_3 = 5\,V$		38,5
Stromaufnahme	I_{CC1}	mA	$U_{CC1} = 16{,}5\,V$		5,5
Stromaufnahme	$-I_{CC2}$		$U_{CC2} = 16{,}5\,V$		30
Referenzspannung mit Last	U_{OREF}	V	$I_{OREF} = 1{,}5\,mA$	9,875	10,125
Linearitätsfehler	F_{LIN}	LSB	$U_{CC1} = -U_{CC2} = -11{,}4\,V$	−1	+1
min. Auflösung ohne Fehlcodes	–	Bit		11	
Unipolaroffset				−4	+4
Nullpunktfehler im Bipolarbetrieb		LSB		−12	+12
Fullscale-Fehler				−21	+21
Umsetzzeit	t_c	µs			40
Betriebsspannungsunterdrückung	SVR	LSB	$13{,}5\,V \leqq U_{CC1} \leqq 16{,}5\,V$ $-13{,}5\,V \geqq U_{CC2} \geqq 16{,}5\,V$	−2	+2
H-Ausgangsspannung	U_{OH}	V	$-I_{16\ldots27} = 0{,}5\,mA$	2,4	–
L-Ausgangsspannung	U_{OL}	V	$I_{16\ldots27} = 1{,}6\,mA$	–	0,4

Die Kennwerte gelten bei $U_{CC1} = -U_{CC2} = 15\,V \pm 0{,}45\,V$, $U_{CC3} = 5\,V \pm 0{,}25\,V$ und $\vartheta_a = 0 \ldots 70\,°C$, falls nicht anders angegeben.

Tafel 3 Ausgewählte Kenngrößen ▲

Tafel 4 Betriebsbedingungen ▼

Kenngröße	Kurzzeichen	Einheit	Kleinstwert	Größtwert
Betriebsspannung für Referenz	U_{CC1}		11,4	16,5
neg. Betriebsspannung	$-U_{CC2}$			
Betriebsspannung für Logikteil	U_{CC3}	V	4,5	5,5
H-Eingangsspannung	U_{IH}		2,0	5,5
L-Eingangsspannung	U_{IL}		0	0,8
Umgebungstemperatur	ϑ_a	°C	0	70

- Pin 9 (Analogmasse) muß direkt mit der Signalquellenmasse verbunden werden. Die Masse der +5 V-Betriebsspannung (Digitalmasse) und die Masse der 15 V-Betriebsspannung (Analogmasse) sollten getrennt geführt und erst nahe am Schaltkreis verbunden werden.
- Der Analogeingangsstrom des A/D-Wandlers ist, bedingt durch das Umsetzverfahren, während des Umsetzens hochfrequent (Taktrate etwa 500 KHz) moduliert.

Die Analogeingangsspannung muß unter diesen dynamischen Lastbedingungen konstant gehalten werden. Es ist eine Signalquelle mit geringer dynamischer Ausgangsimpedanz (bei 500 KHz) erforderlich.

- Der Steuereingang 12/$\overline{8}$ ist nicht TTL-kompatibel und muß daher mit U_{CC3} bzw. Digitalmasse fest verbunden werden.
- Für Offset- und Endwertabgleich sind Spindeleinstellregler mit niedrigem Temperaturkoeffizienten zu verwenden. Bild 2 zeigt einen Vorschlag für die Zusammenschaltung des C 574 C mit dem Mikroprozessor U 880 D. Während der Datenleseoperation beträgt die Buszugriffszeit, gemessen von der L/H-Flanke des CE-Signals bis zum Erscheinen der gültigen Daten an den Datenausgängen, etwa 270 ns.

Die Verzögerungszeit bis Floaten, gemessen von der H/L-Flanke des CE-Signals bis zum völligen Übergang der Datenausgänge in den hochohmigen Zustand, beträgt etwa 300 ns.

⊠ KONTAKT ☎

VEB Halbleiterwerk Frankfurt(Oder),
Abt. EECS, Postfach 379, Frankfurt (Oder), 1200;
Tel. 463264

Termine

Fachtagung Automatisierung 1988 (21. Fachkolloquium Informationstechnik)

WER? Technische Universität Dresden, Sektion Informationstechnik, Wissenschaftsbereich Regelungstechnik und Prozeßsteuerung
WANN? 3. bis 5. Februar 1988
WO? Technische Universität Dresden
WAS?
- Automatisierungskonzepte und -strategien
- Softwareprobleme
- Geräte und Gerätesysteme
- Automatisierung verschiedener Bereiche wie Fertigung, Land- und Forstwirtschaft, Forschung, Energietechnik, Verfahrens- und Verarbeitungstechnik

WIE? Voranmeldungen bis zum 15. 9. 1987 an: Technische Universität Dresden, Sektion 9, Bereich 3, Kennwort Tagung 1988, Mommsenstr. 13, Dresden, 8027; Tel. 4633439. *Prof. Dr. Toepfer*

Mikroprozessorkompatibler D/A-Wandler C 560 D

Henning Zinke
VEB Halbleiterwerk Frankfurt (Oder)

Der D/A-Wandlerschaltkreis C 560 D verarbeitet ein 8 Bit breites Digitalwort in eine analoge Ausgangsspannung. Dabei beträgt der maximale Fehler 0,4 % vom Endwert. Zum D/A-Wandler gehören 8 Eingangsstufen mit pnp-Transistoren, die ein direktes Anlegen eines positiven Logikpegels ermöglichen. Dadurch wird der Betrieb mit einer Versorgungsspannung von Ucc = +5 V ... 15 V gewährleistet. Das nachgeschaltete Eingangs-Latch mit 2 Steuereingängen ermöglicht eine effektive μP-Anpassung des D/A-Wandlers.

Das 8-Bit-R-2R-Widerstandsnetz mit der Referenz und den dazugehörigen Transistoren erzeugt die 8 gewichteten Ströme. Diese werden über die Stromquellenschalter im Ausgangs-OPV in eine Spannung transformiert. Der OPV kann entsprechend der Betriebsspannung und Programmierung des U_OSEL-Anschlusses eine Spannung von U_O = 0 ... +2,5/2,55 V oder U_O = 0 ... +10/10,20 V erzeugen. Mittels einer „Sense"-Leitung (Anschluß UOSEN) wird der exakte Spannungswert am Verbraucher gewährleistet.

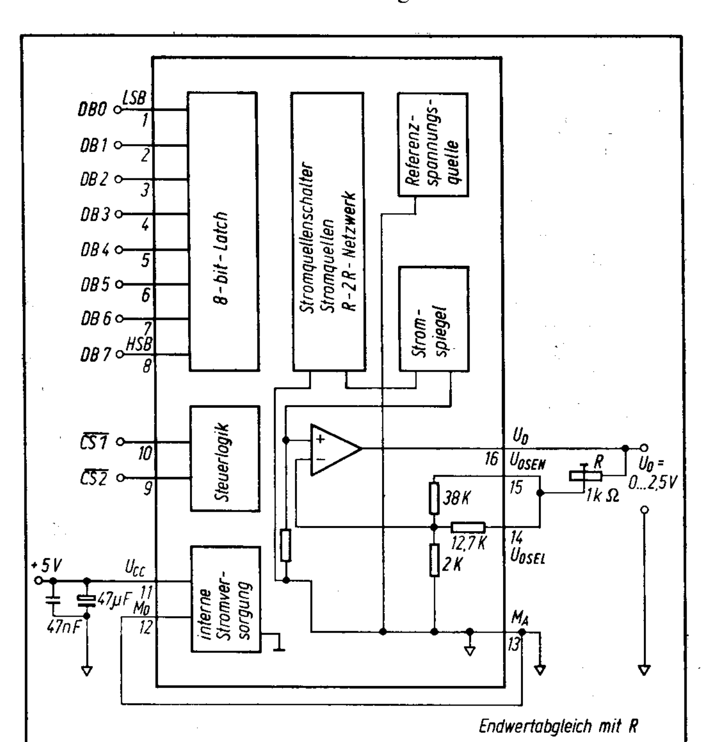


Bild 1 Blockschaltbild des C 560 D mit Beschaltung für U_O = 0... +2,5 V

Tafel 1 Funktionstabelle der Eingangslogik

DB0 bis DB7	$\overline{CS1}$	$\overline{CS2}$	Funktion
H, L	L	L	Latch ist transparent; direkte Umsetzung der Daten
H, L	L ↑	↑ L	Einschreiben der Daten in das Latch (Verriegelung); Umsetzung hat bereits bei $\overline{CS1}$ = $\overline{CS2}$ = L begonnen
X	H X	X H	Latch ist verriegelt; Ausgangsspannung entspricht den zuletzt eingeschriebenen Daten

Tafel 2 Grenzwerte

Kenngröße	Kurzzeichen	Kleinstwert	Größtwert
Betriebsspannung	U_{CC}	0V	18V
Eingangsspannung der Digitaleingänge	U_I	0V	7V
Ausgangsstrom Anschluß 16	I_o	0mA	10mA

Anmerkung: Aufgrund eines Versehens können wir die Tafeln 3 und 4 erst in MP 7/87 veröffentlichen. Wir bitten um Ihr Verständnis. Red.

Terminalsteuerzeichen des PC 1715

Das Terminal des PC 1715 ermöglicht durch die Verwendung von Steuerzeichen eine Ansteuerung von Sonderfunktionen.

Diese Funktionen sind:
- zwei verschiedene Helligkeitsstufen
- zwei verschiedene Zeichensätze
- Unterstreichung ein/aus
- Blinken ein/aus
- normale/inverse Schrift

Die Ansteuerung dieser Funktionen erfolgt durch Einschreiben eines Steuerzeichens in den Bildwiederholspeicher. Dazu ist die Escape-Folge 1 Bh 5 Eh xxh an das Terminal zu senden. Durch diese Folge wird das Byte xxh an der aktuellen Kursorposition in den Bildwiederholspeicher geschrieben und als Steuerzeichen interpretiert. Die jeweils eingestellte Funktion ist bis zum Auftreten eines anderen Steuerzeichens gültig. Die entsprechende Position wird auf dem Bildschirm als ein Leerzeichen dargestellt. Wird diese Position mit einem anderen Zeichen überschrieben, wird die Funktion ausgeschaltet.
Die Verfahrensweise zur Ausgabe der Steuerzeichen ist von der verwendeten Programmiersprache abhängig. Im weiteren wird ein Beispiel für die Programmiersprache des Datenbanksystems REDABAS vorgestellt.
Das Programm besteht aus drei Teilen. Im ersten werden die entsprechenden Bytefolgen als Makro abgespeichert. Im zweiten Teil wird über ein Menü die gewünschte Funktion ausgewählt, die im dritten Programmteil ausgeführt wird. Die entsprechenden Byte-Folgen werden als String in einen Speicher geschrieben, der dann an der entsprechenden Stelle im Programm als Makro aufgerufen wird. Diese Verfahrensweise er-

Bit	Bedeutung	
0	0 = normale Helligkeit,	1 = intensiv
1	0 = Blinken aus,	1 = Blinken ein
2	0 = 1. Zeichensatz	1 = 2. Zeichensatz
3	konstant 0	
4	0 = normale Schrift (wenn Bit 2 = 0)	1 = inverse Schrift
	0 = inverse Schrift (wenn Bit 2 = 1)	1 = normale Schrift
5	0 = unterstreichen aus	1 = unterstreichen ein
6	konstant 1	
7	konstant 0	

Tafel 1 Aufbau des Steuerbytes

```
set talk off
erase

*
*  Initialisierung der Steuerzeichen
*
store 'chr(27)+chr(94)+chr(64)' to NORM
store 'chr(27)+chr(94)+chr(65)' to HELL
store 'chr(27)+chr(94)+chr(67)' to HBLINK
store 'chr(27)+chr(94)+chr(80)' to INVERS
store 'chr(27)+chr(94)+chr(96)' to STRICH
store 'chr(27)+chr(94)+chr(83)' to HBLINV

*
*  Aufbau der Bildschirmmaske
*
@  0,77 say &NORM
@  0, 0 say &STRICH
@  2, 2 say &HELL+' Demonstrationsprogramm '+&NORM
@  3,77 say &NORM
@  3, 0 say &STRICH
@  5, 2 say &HELL+' Vorwaerts mit <X>, zurueck mit <E>,;
                   bestaetigen mit Leertaste '+&NORM
@  6,77 say &NORM
@  6, 0 say &STRICH
@  8, 2 say 'normal, unterstrichen '+&NORM
@ 10, 2 say 'hell '+&NORM
@ 12, 2 say 'hell, blinkend '+&NORM
@ 14, 2 say 'invers '+&NORM
@ 16, 2 say 'hell, invers, blinkend '+&NORM
@ 17,77 say &NORM
@ 17, 0 say &STRICH

*
*  Endlosschleife, mit ESC beenden
*
do while T
        store 'X' to TASTE
        store 6 to X

*
*  Schleife zur Tastenabfrage
*
        do while TASTE#''
              if !(TASTE)='X'
                 store X+2 to X
                 if X=18
                        store 8 to X
                 endif
              endif
              if !(TASTE)='E'
                 store X-2 to X
                 if X=6
                        store 16 to X
                 endif
              endif
              @   X,1 say &INVERS
              @ 21, 0
              wait to TASTE
              @ 22, 0
              @   X,1 say &NORM
        enddo

*
*  Ausfuehrung der gewaehlten Funktion
*
            @ 20, 0
            do case
                 case X=8
                        @ 20,10 say &STRICH+' Das ist normal helle Schrift;
                               mit Unterstreichung '+&NORM
                 case X=10
                        @ 20,10 say &hell+' Das ist helle Schrift '+&NORM
                 case X=12
                        @ 20,10 say &HBLINK+' Das ist blinkende, helle;
                               Schrift '+&NORM
                 case X=14
                        @ 20,10 say &INVERS+' Das ist inverse Schrift '+&NORM
                 case X=16
                        @ 20,10 say &HBLINV+' Das ist helle, inverse blinkende;
                               Schrift '+&NORM
            endcase
enddo
```

Bild 2 Testprogramm zur Demonstration der Terminalfunktionen

leichtert die Eingabe und erhöht die Übersichtlichkeit.

Anmerkung:

Die Steuerzeichen sind hardwareabhängig. Auf den meisten der untersuchten Terminals funktionieren sie so wie beschrieben. Allerdings gibt es Abweichungen. So war bei einem Terminal das Bit für die Helligkeitsumschaltung mit der entgegengesetzten Funktion belegt, bei einem anderen wird statt der Unterstreichung eine Durchstreichung ausgeführt. Bernd Matzke

Übertragung über Lichtleiter

Mit Hilfe der Lichtleitertechnik wurde an der Klinik für Nuklearmedizin der Berliner Charité eine Übertragungsstrecke für eine Szintillationskamera eingerichtet. Dieses Diagnosegerät ist über eine Entfernung von 650 Metern mit einem Bildverarbeitungssystem verbunden, das mit einer Übertragungsrate von 5 MBits pro Sekunde arbeitet. Mit dieser Lösung schuf ein Kollektiv der Klinik gemeinsam mit Mitarbeitern des Zentrums für Forschung und Technologie des Kombinats Elektroapparatewerke Berlin die Möglichkeit, den angeschlossenen Prozeßrechner jetzt für eine zweite örtlich entfernte Szintillationskamera zu nutzen. ADN

Labormuster vom 4-Megabit-Speicherchip

Siemens stellte erste Labormuster eines integrierten Halbleiter-Bausteins vor, der mehr als vier Millionen Bit speichern kann. Der 1-MBit-DRAM soll 1987 in die Massenfertigung gehen, 1989 der 4-MBit-DRAM. Von diesem Baustein der 4 194 304 Bit auf 91 mm^2 (6,5 mm × 14 mm) speichern kann, liegen jetzt die ersten Chips vor. Die wesentliche Neuheit in diesem Speicher ist die sogenannte Trench- oder Grabenzelle, bei der ein nur 1 μm breiter Graben 4 μm tief in das Silizium geätzt wird.
Für die mehr als 4 Millionen Speicherzellen auf einem Silizium-Chip sind 450 einzelne Prozeßschritte in CMOS-Technik erforderlich. Gegenüber dem 1-MBit-DRAM (54 mm^2) wird die Speicherkapazität vervierfacht, die Chipfläche mit 91 mm^2 jedoch nicht einmal verdoppelt. Die Trenchzelle, die je eine binäre Informationseinheit speichert, benötigt nur 5 μm^2 Scheibenfläche. Insgesamt kann der Chip den Inhalt von etwa 250 Schreibmaschinenseiten speichern. Die 4-MBit-Technologie wird in enger Zusammenarbeit mit dem niederländischen Philips-Konzern entwickelt.

K 1520-Mikrorechnerplatz mit Kassettenmagnetband

Dr. Joachim Mertins, Johannes Münch
VEB Elektroprojekt und Anlagenbau Berlin, Zentrum für Forschung und Technologie

Es wird eine K1520-Mikrorechnerkonfiguration vorgestellt, die ausschließlich auf Robotron-Baugruppen basiert /1/, /2/ und ein kassettenmagnetbandorientiertes Betriebssystem (KAMOS) verwendet. Als Zentraleinheit wird K2521 verwendet. Die Speichergröße beträgt 64 KByte RAM. Die wichtigsten Dienstprogramme stehen auf EPROM sofort zur Verfügung. Die Software und die Dokumentation des Systems sind nachnutzbar. Der Mikrorechner-Arbeitsplatz zeichnet sich durch schnellen und zuverlässigen Zugriff zu den Kassetten aus.

Monitorprogramm

Das Monitorprogramm (9 KByte) realisiert Grundfunktionen der Rechnerarbeit:
- Programmstart, Start von System- und Dienstprogrammen von EPROM (Phasenlader für Kassette, Editor, Assembler, Kassettenkopierprogramm, Anzeige Kassettenverzeichnis) durch Funktionstasten
- RAM-Speicher lesen und schreiben
- Maschinenkode-Lochstreifen lesen und stanzen
- Breakpoint, Registeranzeige, Dump, Prüfsummenbildung
- Füllen und Umladen von Speicherbereichen
- Unterprogramme für Nutzung durch Anwender, die Bildschirm, Tastatur, Drucker und Lochbandperipherie bedienen
- Programmieren und Lesen EPROM (1 K bzw. 2 K)

Kassettenmagnetbandarbeit und Dateikonzept

Durch Minimierung von Umspul- und Positionieroperationen, eine große Blocklänge (256 Byte), sequentielle Abspeicherung, Verzicht auf ein separates Dateiverzeichnis (Directory) auf Kassette, labelorientierten Zugriff zu den Dateien über Bandmarken sowie Arbeit mit zwei Kassettenlaufwerken (Quell- und Zielkassette) wird die Arbeit der Dienstprogramme beschleunigt.
Je Kassettenseite sind bis 256 Dateien möglich, die durch Bandmarken getrennt sind und aus sequentiellen Sätzen bestehen. Als Dateitypen sind Textdateien (Quellprogramm/freies Format) und Binärdateien (ladbare Programme mit Ladeadresse im Maschinenkode Z80/U880) möglich.
Der Phasenlader ermöglicht das Laden von Binärdateien von Kassette in den RAM-Speicher und den Programmstart für abarbeitbare Programme, die Ausgabe von RAM-Bereichen auf Kassette und allgemeine Servicefunktionen für Kassettengeräte (Vorspulen/Rückspulen bis n-te Bandmarke, Rückspulen zum Bandanfang, Zuordnung logische und physische Geräte).
Für die Nutzung durch Anwender stehen Unterprogramme zur Bedienung der Kassettengeräte und zum Lesen und Schreiben von sequentiellen Dateien zur Verfügung.

Software

Falls kein Kassettenmagnetbandgerät und nur Lochbandtechnik vorgesehen ist, wird eine Programmentwicklung und -testung über Lochband-Editor und -Assembler unterstützt.
Die wichtigsten Dienstprogramme, die Kassettenarbeit unterstützen, sind: Editor, Assembler Z80/U880 (Absolut-Assembler für Erzeugung abarbeitbarer Programme), Kopierprogramm für eine oder mehrere Dateien in beliebiger Reihenfolge oder für gesamte Kassetten, Reassembler Z80/U880 mit 2 KByte Speicherbedarf, BASIC-Interpreter mit 12 KByte Speicherbedarf (bzw. TINY-BASIC-Interpreter mit 14 KByte), ein FORTH-Interpreter sowie Unterstützung der Textverarbeitung (mehrfacher Druck einer Datei bzw. mehrerer Dateien ohne Bienereingriff). Außerdem wird die Kopplung mit einem SKR-Rechner unterstützt (Übertragung von Dateien).

Kompatibilität zum Bürocomputer A 5120/5130

Auf dem Bürocomputer wurde unter den Betriebssystemen UDOS und CP/M (Version CPA) ein KAMOS-kompatibles Kassettenbediensystem installiert. Mit dessen Hilfe ist ein Dateitransfer von Quell- und Binärdateien zwischen Diskette und Kassette möglich. Das Kassettenformat ist dabei zwischen den Betriebssystemen UDOS, CP/M und KAMOS kompatibel /3/.

Literatur

/1/ Mertins, J.: Mikrorechnerarbeitsplatz aus K 1520-OEM-Baugruppen. Sozialistische Rationalisierung in der Elektrotechnik/Elektronik, 14 (1985) 2, S. 39–43

/2/ Mertins, J.; Münch, J.: Betriebssystem KAMOS für K 1520-OEM-Konfiguration. Sozialistische Rationalisierung in der Elektrotechnik/Elektronik 15 (1986) 5, S. 105–108

/3/ Kassettenmagnetband für Bürocomputer unter UDOS. rechentechnik/datenverarbeitung 22 (1985) 9, S. 29

⊠ KONTAKT ☏

VEB Elektroprojekt und Anlagenbau Berlin, Zentrum für Forschung und Technologie, Abt. RC4, Rhinstraße 100, Berlin, 1140, Tel. 5509531 App. 29

MP-Börse

Nutzerhandbuch BDS–C

An unserer Einrichtung wurde ein Nutzerhandbuch für die Sprache C (BDS–C, Vers. 1.50) erarbeitet. Im Teil 1 wird der Weg von der Erstellung eines Quell-Programmes bis zum lauffähigen Maschinen-Programm beschrieben. Der Teil 2 enthält eine vollständige Aufstellung der etwa 200 Funktionen umfassenden Library. Die Möglichkeit der Verwendung von Long-integer- bzw. Floating-point-Werten wird im Teil 3 erläutert. Der Teil 4 beschreibt die Arbeitsschritte zur Erzeugung von Maschinenprogrammen, die nicht unter dem Betriebssystem CP/M laufen sollen. Im Teil 5 werden einige Beispiele für C-Programme demonstriert.

Interessenten wenden sich an die Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Wissenschaftsbereich Informationsverarbeitung, Richard-Wagner-Str., Warnemünde, 2530, Tel.: 57345/Koll. Chmiela.

Lehrsystem für die Ausbildung am Computer

Mit dem Lehrsystem LKO 1520/2 ist es möglich, Grundkenntnisse der Programmierung und Nutzung eines Computers im Dialog am Computer zu erlernen. Es sind Lehrprogramme für BASIC, POWER, REDABAS, TP und Supercalc vorhanden, die das Selbsterlernen des Umgangs mit der Software bei geringen Vorkenntnissen ermöglichen. LKO 1520/2 ist einsatzfähig auf PC 1715, A 5120/30 (Betriebssystem SCP) und KC 85/2 (3). Für die eigene Erarbeitung weiterer Lehrprogramme steht das Autorensystem AUT 1520 für die genannten Rechnertypen zur Verfügung.

Interessenten wenden sich an die Betriebsschule des VEB Plauener Gardine, Dobenaustraße 80, Plauen, 9900, Tel. 27051, Dr. Eckardt.

Programmierung in C

Teil V

Dr. Thomas Horn
Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden

Im vorletzten Teil der Artikelreihe zu C wird das Ein- und Ausgabekonzept der Programmiersprache C behandelt, das kein integrierter Bestandteil der Sprachdefinition ist, sondern über eine umfangreiche Funktionsbibliothek realisiert wird. Es gibt ein Minimum an E/A-Funktionen, die als Quasi-Standard von allen C-Systemen realisiert werden. Diese Funktionen werden im wesentlichen im vorliegenden Beitrag behandelt. Sie garantieren auch eine Portabilität der C-Programme unter den verschiedenen Rechnern und Betriebssystemen. Weiterhin werden einige Hinweise zur Verarbeitung von C-Programmen gegeben. Das Prinzip wird hier im wesentlichen allgemeingültig dargestellt, wobei unter den einzelnen Betriebssystemen im Detail, vor allem bei den Kommandos und Filespezifikationen, geringfügige Abweichungen auftreten können. Über diese Besonderheiten sollte man sich immer informieren, wenn man mit einem bestimmten C-Sprachverarbeitungssystem unter einem konkreten Betriebssystem zu arbeiten beginnt. Der letzte Teil dieser Artikelreihe wird dann einige komplexe Programmbeispiele behandeln, an denen auch die Programmierungstechnik in C verdeutlicht werden soll.

9.5. Zeilennumerierung

#line *constant identifier*

definiert die Zeilennummer und den Namen eines Quellprogramms. Die Präprozessoranweisung wird zur besseren Identifikation von Fehlermeldungen bei C-Programmen benutzt, die durch zahlreiche Fileeinfügungen unübersichtlich geworden sind. Die Anweisung ermöglicht, daß sich die Fehlermeldungen auf die Zeilennummern verschiedener Quellfiles beziehen können. Wenn in der Anweisung kein Identifikator spezifiziert ist, so wird der letzte Identifikator weiter benutzt.

Beispiel:
#line 1 TESTO
Die folgende Zeile des Files **TESTO** bekommt die Zeilennummer 1.

Beispiel:
Wenn der Generierungsparameter **TIME** ungleich 0 ist, soll die Struktur **datum** mit der Komponente Zeit generiert werden, anderenfalls soll die Struktur keine Komponente Zeit definieren.

```
#define TIME 1
...
#if TIME
struct datum
{int tag;
char *monat;
char jahr[2];
int zeit[2];
#else
struct datum
{int tag;
char *monat;
char jahr[2];}
#endif
```

Die generierungsabhängige Struktur **datum** kann auch vereinfacht wie folgt definiert werden:

```
struct datum
{int tag;
char *monat;
char jahr[2];
#if TIME
int zeit[2];
#endif
}
```

Die zweite Schreibweise ist der ersten äquivalent.

10. Ein- und Ausgabefunktionen

Die Ein- und Ausgabeorganisation ist, wie bereits erwähnt, kein Bestandteil der Sprachdefinition der Programmiersprache C. Sie wird grundsätzlich über Funktionen der Standard-Unterprogrammbibliothek (CSLIB.OLB oder C.OLB) realisiert, die größtenteils Assemblerunterprogramme zur Anpassung an das Betriebssystem des jeweiligen Rechners enthält. Durch die Standard-Unterprogramme der Unterprogrammbibliothek wird im wesentlichen die Portabilität der C-Programme abgesichert. Diese Unterprogramme (Funktionen) werden nachfolgend behandelt.

10.1. Prinzipien der Fileverarbeitung

Unter einem File wird eine logisch zusammenhängende Menge von Datensätzen verstanden, die nach einem bestimmten Organisationsprinzip auf externen Datenträgern, wie Disketten, Plattenspeicher, Magnetbandkassetten usw., gespeichert werden. Vielfach wird in der Literatur der früher stärker benutzte Begriff Datei dem Begriff File gleichgesetzt, wobei aber der Begriff Datei vom Organisationsprinzip des Files (Filestruktur) abstrahiert. Files werden auf den externen Datenträgern durch Filespezifikationen identifiziert. Eine Filespezifikation besteht im allgemeinen aus einem Gerätenamen (*dev*:), einem Verzeichnisnamen (*dir*/oder **[***dir***]**), einem Filenamen (*fname*) und einem Filetyp (*.ftyp*):

dev:*dir*/*fname*.*ftyp*
oder
dev:**[***dir***]***fname*.*ftyp*;*version*

In CP/M-kompatiblen Betriebssystemen entfällt generell die Angabe eines Verzeichnisses. In MS-DOS- bzw. UNIX-kompatiblen Betriebssystemen wird das Verzeichnis durch einen Schrägstrich vom Filenamen getrennt. Es können auch – durch Schrägstriche getrennt – noch Unterverzeichnisse (*Subdirectories*) spezifiziert werden. Im Betriebssystem OS-RW auf SKR-Anlagen wird das Verzeichnis generell in eckigen Klammern spezifiziert, und nach dem Filetyp kann noch eine Versionsangabe (;*version*) folgen. Beim Zugriff auf das Standardgerät, das Standardverzeichnis des Nutzers und die letzte Versionsnummer eines Files können die entsprechenden Angaben in der Filespezifikation entfallen, so daß nur der Filename und der Filetyp angegeben werden müssen.
Für nicht filestrukturierte Geräte, wie Drucker, Terminals usw., müssen die entsprechenden Gerätenamen spezifiziert werden, wobei die Angabe von Filename und Filetyp entfallen kann, z. B. für den Drucker in Abhängigkeit vom Betriebssystem **PRT:**, **LST:** oder **LP:** usw.
In Betriebssystemen werden sequentielle Files und Direktzugriffsfiles unterschieden.
Sequentielle Files können auf allen Geräten ein- oder ausgegeben werden, wie Diskettenspeicher, Magnetbandgeräte,

Terminals, Drucker usw. Direktzugriffsfiles können nur auf Geräten mit wahlfreiem Zugriff, wie Diskettenspeicher und Plattenspeicher, ein- und ausgegeben werden. Da die sequentiellen Files von grundlegender Bedeutung sind und betriebssystemunabhängig realisiert werden können, „standardisiert" die Standard-Unterprogrammbibliothek nur die Verarbeitung sequentieller Files. Die Verarbeitung von Direktzugriffsfiles ist im wesentlichen betriebssystemabhängig und soll deshalb nur als Beispiel für das Betriebssystem UNIX zum Schluß dieses Abschnitts skizziert werden.
Die Verarbeitung von Files schließt folgende Funktionen ein:
- Eröffnen des Files (*open*)
- Ein- oder Ausgabe der Datensätze (*read* oder *write* bzw. *get* oder *put*)
- Schließen des Files (*close*).

Für die Fileeröffnung wird bei sequentiellen Files die Funktion **fopen** verwendet. Bei der Fileeröffnung wird vom Laufzeitsystem ein Filedeskriptorblock (FDB) eingerichtet. Bei der Fileverarbeitung mit den Ein- und Ausgabefunktionen, wie **getc**, **putc**, **fgets**, **fputs** usw., wird auf den FDB über einen Filepointer *fp* Bezug genommen. Nach der Fileverarbeitung wird mit der Funktion **fclose** das File wieder geschlossen und der FDB gelöscht.
Zur Vereinfachung der Anwendung der Fileverarbeitung existiert ein File *STDIO.H*, das die Strukturdefinition für den FDB enthält. Die Bezugnahme erfolgt über den Strukturentyp **FILE**, mit dem beliebige Filepointer eingerichtet werden können. Voraussetzung ist, daß das File *STDIO.H* mit der Anweisung **#include <stdio.h>** am Programmanfang in das C-Programm eingefügt wurde. (Das File *STDIO.H* enthält außerdem noch Definitionen für die Symbole **FALSE** (0), **TRUE** (1), **NULL** (0), **EOF** (−1) und **EOS** (\0).)
Die Fileeröffnung kann dann wie folgt durchgeführt werden:

fp=**fopen** (*name*, *mode*);

wobei
name
die Filespezifikation angibt, die entsprechend den Festlegungen des verwendeten Betriebssystems aufgebaut sein muß,
mode
die Zugriffsart spezifiziert:
„**r**" – Lesezugriff,
„**w**" – Schreibzugriff,
„**a**" – Schreibzugriff zum Anfügen an das File.

Dem Filepointer wird bei fehlerfreier Fileeröffnung die Adresse des FDB zugewiesen. Bei fehlerhafter Fileeröffnung wird der Filepointer *fp* auf Null gesetzt. Diese Bedingung kann mit dem Symbol **NULL** getestet werden.
Der Fileabschluß erfolgt mit:

fclose (*fp*);

Bei normaler Programmbeendigung wird jedes noch offene File automatisch geschlossen.
Durch das Laufzeitsystem werden drei Filepointer für das Standardgerät des Nutzers, das Bedienterminal (**CON:** oder **TI:**), automatisch eingerichtet:

stdin – als Standardeingabegerät
stdout – als Standardausgabegerät
stderr – als Standardfehlerausgabegerät.

Diese Filepointer brauchen nicht definiert und die Files nicht eröffnet werden (sind in *STDIO.H* definiert).

Beispiel:
```
#include <stdio.h>
main ()
{
FILE *fp1,*fp2;
if((fp1=fopen("TEST.DAT","r"))
          ==NULL) goto ENDE;
if((Fp2=fopen("LP:","w"))
          ==NULL) goto ENDE;
...
fclose(fp1); fclose(fp2);
ENDE:
}
```

10.2. Ein- und Ausgabe von Zeichen

Die zeichenweise Ein- und Ausgabe kann mit den Funktionen **getc** und **putc** durchgeführt werden:

c=**getc**(*fp*);
putc(*c*,*fp*);

wobei *c* die Variable (im Byteformat) ist, die ein- oder ausgegeben werden soll. Wenn bei einer Eingabeanweisung kein Zeichen mehr zur Verfügung steht, so wird der Variablen *c* der Wert **−1** (**EOF** – **E**nd **O**f **F**ile) zugewiesen, der mit dem Symbol **EOF** ausgetestet werden kann. Zur Vereinfachung der Nutzung des Standardein- bzw. -ausgabegerätes stehen zwei weitere Funktionen zur Verfügung: *c*=**getchar()**;
entspricht: *c*=**getc(stdin)**;
putchar(*c*);
entspricht: **putc**(*c*,**stdout**);

Beispiel:
Angenommen, im Beispiel 10.1 soll das File **"TEST.DAT"** auf **"LP:"** protokolliert werden, so sind folgende Anweisungen zu ergänzen:

```
char c;
while((c=getc(fp1))!=EOF)
putc(c,fp2);
```

10.3. Ein- und Ausgabe von Zeilen

Die Ein- und Ausgabe von Zeilen kann mit den Funktionen **fgets** und **fputs** erfolgen:

s=**fgets**(*line*,*lmax*,*fp*);
fputs(*line*,*fp*);

wobei
line das Feld mit der Zeile,
lmax die maximal einzugebende Zeilenlänge und
s Adresse des Satzes oder **NULL** (bei Fileende) ist.
Es ist zu beachten, daß das letzte Zeichen einer Zeichenkette immer ein Null-Zeichen (**EOS** – **E**nd **O**f **S**tring) ist; das heißt, wenn *lmax* mit 80 Zeichen vorgegeben ist, beträgt die tatsächliche maximale Zeilenlänge nur 79 Zeichen. Genauso fordert **fputs**, daß die auszugebende Zeile mit dem Steuerzeichen **EOS** abgeschlossen ist.
Zur Vereinfachung der Arbeit mit dem Standardgerät des Nutzers stehen zwei weitere Funktionen zur Verfügung:
gets(*line*)
entspricht: **fgets**(*line*,*lmax*, **stdin**)
puts(*line*)
entspricht: **fputs**(*line*,**stdout**)

Beispiel
```
#include <stdio.h>
main()
{
char zeile[80]; FILE*fp1;
if((fp1=fopen("TEST.C","r"))
==NULL)
error("File can't be opened");
while(fgets(zeile,80,fp1))
fputs(zeile,stdout);
fclose(fp1);
}
```

10.4. Formatierte Ein- und Ausgabe

Für die formatierte Ein- und Ausgabe stehen die zwei Funktionen **fprintf** und **fscanf** zur Verfügung:

fprintf(*fp*,*format*,*arg1*,*arg2*, . . .)
fscanf(*fp*,*format*,*arg1*,*arg2*, . . .)

wobei
format – die Formatbeschreibung und
argi – die Argumente angeben.

Die Formatbeschreibung gibt den Aufbau des externen Datensatzes an, wobei spezifiziert wird, an welcher Stelle sich welche Argumente in welchem Datenformat befinden sollen. Die Datenformate der Argumente werden durch Formatelemente beschrieben, die mit ge-

wöhnlichen Textzeichen beliebig gemischt werden können. Bei der Ausgabe geben die Formatelemente somit an, an welcher Stelle welche Argumente in welchem Format eingefügt werden sollen. Bei der Eingabe geben die Formatelemente an, wie die Eingabezeichenkette zu interpretieren ist.
Die Formatelemente beginnen mit einem Prozentzeichen **%**. Das Format wird durch ein nachfolgendes Zeichen spezifiziert:

d dezimale Darstellung des Arguments (**int**)
u dezimale Darstellung des Arguments ohne Vorzeichen (nur Ausgabe) (**int**)
o oktale Darstellung des Arguments (**int**)
x hexadezimale Darstellung des Arguments (**int**)
c das Argument ist ein Zeichen (**char**)
s das Argument ist eine Zeichenkette (**char**)
f Eingabe eines Gleitkommaarguments (**float** oder **double**)
Ausgabe eines Gleitkommaarguments in der Punktdarstellung *[−]mmm.nnnnnn*.
e Ausgabe eines Gleitkommaarguments in der Exponentendarstellung *[−]m.nnnnnn***E***[+−]xx*.
g Ausgabe eines Gleitkommaarguments in der Form **%e** oder **%f**, je nachdem, welches Format kürzer ist.

Soll in dem Text ein %-Zeichen ausgegeben werden, so muß **%%** spezifiziert werden.
Bei der Ausgabe kann zwischen dem %-Zeichen und dem Formatzeichen eine ausführliche Druckformatspezifikation stehen:

– Ein *Minuszeichen*, das angibt, daß das umgewandelte Argument in seinem Feld linksbündig ausgerichtet wird.
– Eine *Zahl*, die die minimale Feldbreite definiert. Erfordert das Argument nicht die volle Feldbreite, so wird das Feld mit Leerzeichen aufgefüllt. Beginnt die Zahl, die die Feldbreite spezifiziert, mit einer Null, so wird das Feld mit Nullen aufgefüllt.
– Ein *Punkt* mit einer *Zahl*, die eine maximale Feldbreite definiert. Bei **float** und **double** gibt diese Zahl die Stellen nach dem Dezimalpunkt an.
– Ein *Buchstabe* **l**, der angibt, daß das zugehörige Argument im **long**-Format (bzw. **double**-Format) spezifiziert ist.

Bei der Eingabe werden Leerzeichen, Tabulatoren und Zeilenvorschubzeichen in der Formatbeschreibung ignoriert.
Sonstige Zeichen (außer **%**) müssen in der Eingabezeile auftreten. Die Formatelemente legen den Datentyp der Argumente fest. Zwischen dem **%**-Zeichen und dem Formatzeichen können stehen:
– Ein Stern (*), der angibt, daß das Eingabefeld übersprungen wird. Eine Wertzuweisung erfolgt nicht.
– eine Zahl, die die Feldbreite des Eingabefeldes spezifiziert
– ein Buchstabe **l** vor den Formatzeichen **d**, **o** oder **x** zeigt an, daß das entsprechende Argument im **long**-Format spezifiziert ist. Vor dem Umwandlungszeichen **e** oder **f** weist **l** auf das **double**-Format hin.

Bei der Eingabe ist zu beachten, daß die Argumente Zeiger sein müssen. Bei Feldvariablen ist das automatisch gegeben. Bei einfachen Variablen ist, falls kein Zeiger vorhanden, die Adresse über den Adreßoperator **&** zu spezifizieren.
Zur Vereinfachung der Nutzung des Standardein- und -ausgabegerätes stehen zwei weitere Funktionen zur Verfügung:

printf(*format,arg1,arg2,...***)**
scanf(*format,arg1,arg2,...***)**

Diese beiden Funktionen entsprechen den Funktionen **fprintf** und **fscanf**, mit der Ausnahme, daß kein Filepointer angegeben wird.

Beispiel:
```
main()
{
int n;
float s;
char name[40];
scanf("%s %d %f", name,&n,&s);
printf("\nName:%-20s Nr. %-4d
        Gehalt: %8.2f M", name,n,s);
printf("\n\nEnde
des Programms\n");
}
```

Mit **scanf** wird eine Eingabe angefordert. Es wird folgende Zeichenkette eingegeben:

Meier 713850.6<CR>

Auf dem Terminal erscheinen folgende Ausgaben:

Name: Meier Nr.713 Gehalt: 850,60 M

Ende des Programms

10.5. Formatumwandlung im Speicher

Analog zu den Funktionen **fprintf** und **fscanf** existieren zwei weitere Funktionen, **sprintf** und **sscanf**, die die gleichen Formatumwandlungen vornehmen, aber die Zeichenketten nicht ein- und ausgeben, sondern die Zeichenketten im Speicher ablegen bzw. im Speicher voraussetzen. Diese Funktionen werden wie folgt aufgerufen:

sprintf(*string,format,arg1,arg2,...***)**
sscanf(*string,format,arg1,arg2,...***)**

wobei
string die Zeichenkette spezifiziert, die bei **sprintf** das Ergebnis der Formatwandlung aufnehmen soll bzw. bei **sscanf** die umzuwandelnde Zeichenkette enthält,
format die Formatbeschreibung (siehe 10.4.) und
argi die Argumente (siehe 10.4.) angeben.

10.6. Fehlerausgaben, Programmbeendigungen und Zusatzfunktionen

Für Fehlerausgaben und Programmbeendigungen wird meistens die Funktion

error("*string***")**

benutzt, wobei *string* die Fehlernachricht spezifiziert. Die Fehlernachricht wird auf **stderr** ausgegeben. Die Files werden ordnungsgemäß abgeschlossen und das Programm abgebrochen. (Für den Parameter "*string*" kann auch eine Parameterspezifikation wie in der **printf**-Funktion stehen.)
Ein Programm kann auch mit der Funktion **exit(***n***)** abgebrochen werden, wobei *n* einen Fehlerkode spezifiziert. Eine Null (0) zeigt eine fehlerfreie Ausführung an. Auch bei **exit** werden alle Files ordnungsgemäß abgeschlossen. Vor **exit** kann eine Aufbereitung einer Fehlermitteilung mit **fprintf** erfolgen.

Beispiel:
```
fprintf(stderr,"\nFile%s kann
nicht eröffnet werden
\n", name);
exit(1);
```

Bei der Eingabe von Zeichen kann mit der Funktion

ungetc(*c,fp***)**

das jeweils letzte Zeichen wieder in das File zurückgeschrieben und somit das File modifiziert werden. Hierbei haben die Parameter folgende Bedeutung:
c – zurückzuschreibende Variable
fp – Filepointer des zu modifizierenden Files.

10.7. Elementare E/A-Funktionen

Die elementaren E/A-Funktionen gestatten im Betriebssystem UNIX die unmittelbare Verarbeitung von beliebigen physischen Datenträgern im sequentiellen Zugriff und Direktzugriff. Der Trans-

fer wird dabei direkt zwischen dem E/A-Puffer und dem E/A-Gerät organisiert. Folgende Funktionen stehen zur Verfügung:

fd=**open(***name,rwmode***)**:
Eröffnen eines Files
fd=**creat(***name,pmode***)**:
Erstellen eines neuen Files
na=**read(***fd,buf,n***)**:
Lesen eines Blockes
na=**write(***fd,buf,n***)**:
Schreiben eines Blockes
close(*fd***)**:
Schließen eines Files
unlink(*name***)**:
Löschen eines Files
lseek(*fd,offset,position***)**:
Positionieren auf ein Byte
wobei
fd – Filedeskriptornummer für das File,
name – Filespezifikation,
rwmode – Zugriffsart:
0 – nur Lesezugriff
1 – nur Schreibzugriff
2 – Lese- und Schreibzugriff
pmode – Fileschutzfestlegung
(neun Bits):

rew	*rew*	*rew*
Eigentümer	Gruppenmitglieder	Übrige

r =**1** – Lesen erlaubt
e =**1** – Ausführen erlaubt
w=**1** – Schreiben erlaubt
buf – E/A-Puffer
n – Blockgröße (in der Regel 512 Byte)
na – aktuelle Blockgröße in Byte (tatsächlich ein- oder ausgegebene Byteanzahl)
offset – Nummer des Bytes, auf das positioniert werden soll. Diese Angabe muß im **long**-Format spezifiziert werden!
position – Fileposition als Basis für *offset:*
0 = Fileanfang
1 = aktuelle Fileposition
2 = Fileende

Der Parameter *fd* ist eine Filedeskriptornummer, die nicht mit dem Filepointer *fp* verwechselt werden darf. *fd* ist immer eine positive Integerzahl. Die Nummern **0**, **1** und **2** sind für die Files **stdin**, **stdout** und **stderr** vergeben. Werden weitere Files eröffnet, so werden weitere Nummern vergeben. Wenn ein File nicht eröffnet werden kann, so wird *fd* auf **−1** gesetzt.
Der Parameter *na* wird auch zur Fehlererkennung verwendet. Ein Wert **−1** zeigt einen Ein- bzw. Ausgabefehler und ein Wert **0** bei Eingabefiles das Fileende an.

Beispiel:
```
#include<stdio.h>
main()
{
char block[512];
int file1,file2;
if ((file1=open("TEST.TXT",0))<0)
error ("File 1 nicht vorhanden");
if ((file2=creat
("TEST.DOC",0766))<0)
   error ("File 2 kann nicht
   angelegt werden");
while ((n1=read(file1,block,
512))!=0)
{if (n1<0) error ("Eingabefehler");
if ((n2=write(file2,block,n1)!=n1)
error ("Ausgabefehler");}
close (file1); close (file2);
}
```

11. Methodik der Programmentwicklung

Unter fast allen Betriebssystemen erfolgt die Verarbeitung von Quellprogrammen, die in der Programmiersprache C geschrieben sind, nach dem Prinzip der zweistufigen Programmentwicklungsmethodik (Bild 1). Dieses Prinzip schließt die Verwendung eines Zwischenkodes, des sogenannten Objektkodes, ein. Jeder Quellmodul *fname*.**C** wird in der ersten Verarbeitungsstufe durch den C-Compiler **CC** in einen Objektmodul
fname.**OBJ** übersetzt. Unter einem Quellmodul wird allgemein eine Übersetzungseinheit des Compilers verstanden. Der C-Compiler läßt zu, daß ein Quellmodul eine oder mehrere komplette Funktionen enthält. Der Objektmodul enthält im wesentlichen das in den Maschinenkode des Zielrechners übersetzte C-Programm sowie Steuerinformationen über die Definition globaler Symbole und die Bezugnahme auf globale (externe) Symbole. Während der Übersetzung eines Quellmoduls kann der C-Compiler auf das File *STDIO.H* mit der Definition des Filedeskriptorblockes und der Filepointer für die Standardgeräte Bezug nehmen. Ebenso können beliebige andere Quellfiles mit der **#include**-Anweisung vom C-Compiler in den Quellmodul eingefügt werden.
In der zweiten Verarbeitungsstufe werden die getrennt übersetzten Objektmoduln durch den Programmverbinder (**LINK** – Linker oder **TKB** – Taskbuilder) zu einem ausführbaren Maschinenkodeprogramm (*fname*.**EXE** oder *fname*.**TSK**) verbunden. Dabei nimmt der Programmverbinder auch auf die Objektmodulbibliothek des C-Laufzeitsystems (**CSLIB.OLB** oder **C.OLB**) und gegebenenfalls auch auf private Objektmodulbibliotheken Bezug. Aus den Objektmodulbibliotheken werden nur die Objektmoduln verwendet, die benötigt werden, um noch offene Bezugnahmen auf externe (globale) Symbole auflösen zu können. An den bezugnehmenden Stellen in den verschiedenen Objektmoduln ergänzt der Programmverbinder die absoluten virtuellen oder physischen Adressen der externen (globalen) Symbole. Das Ergebnis der Programmverbindung ist der ausführbare Lademodul, der alle benötigten Hilfsfunktionen beinhaltet und keine offenen Bezugnahmen auf externe (globale) Symbole mehr enthält.
Die Aufbereitung der C-Quellmoduln erfolgt mit einem Standard-Editor des jeweiligen Betriebssystems. Da der C-Compiler keine Drucklisten des Quellmoduls erstellt, sondern nur eine Fehlerliste ausgibt, müssen für die Ausgabe der Druckliste des Quellmoduls die Betriebsfunktionen wie **TYPE**, **PRINT** oder **PIP** genutzt werden.
Der Programmverbinder erstellt bei Bedarf eine Druckliste, die die eingebundenen Objektmoduln und die Hauptspeicheraufteilung ausweist.
Der Start des Lademoduls erfolgt mit dem **RUN**-Kommando des Betriebssystems. Spezielle Testhilfen werden für den Test von C-Programmen nicht zur Verfügung gestellt. Zweckmäßigerweise bedient man sich für eine Testunterstützung der **printf**-Funktion, mit der man für Testzwecke die Zustände von internen Variablen an bestimmten Stellen des Algorithmus ausgibt. Später werden diese **printf**-Anweisungen entweder wieder aus dem Programm gelöscht oder in Kommentare umgewandelt.
Für die Verarbeitung eines C-Programms unter einem bestimmten Betriebssystem muß man sich in jedem Fall mit den entsprechenden Anweisungen zur Benutzung der verschiedenen Betriebssystemkomponenten, des C-Compilers und des Programmverbinders vertraut machen.

Bild 1 Methodik der Programmentwicklung

Computerkopplung KC 85/3 – PC1715 über V.24-Interface

Klaus-Dieter Kirves,
Karsten Schiwon
VEB Mikroelektronik „Wilhelm Pieck“
Mühlhausen

Mit der Bereitstellung des Moduls M003-V24 für die Kleincomputer KC 85/2 und KC 85/3 ist nicht nur eine Möglichkeit für die Kleincomputer geschaffen worden, Daten auf Drucker auszugeben, sondern auch die einer Kopplung von Kleincomputern untereinander oder zu anderen Computern, wie z. B. dem PC 1715. Dabei ist eine sinnvolle Ergänzung der Möglichkeiten beider Computer denkbar, wie das Erfassen von Daten auf Kleincomputern und deren rechnerisches Auswerten auf dem Personalcomputer oder auch das grafische Darstellen errechneter Werte auf dem Kleincomputer. Weiterhin stehen durch eine Kopplung die Diskettenlaufwerke der PC 1715 auch den Kleincomputern als externe Speicher zur Verfügung.

Tafel 1 Belegung des Verbindungskabels

Signalleitung am PC	Anschluß am PC 1715 Steckverbinder 26polig	Anschluß KC 85/3 Diodenbuchse
Masse	A1,B1,B2	2
RxD	B4	3
DTR	B8	4
TxD	A3	1
CTS	B6	5
DCD	A9 – Brücke zu B8	

Tafel 2 Eingestellte Parameter

Device:	PC 1715 V.24
Übertragungsrate:	1200 Baud
Bit pro Zeichen:	8
Anzahl der Stoppbits:	1
Protokollart:	DTR
Parität:	keine

So, wie die Anwendungs- und Einsatzgebiete der Computer in den verschiedenen Fällen starke Unterschiede haben, werden auch die Aufgaben einer Kopplung von Computern stark differieren. Deshalb kann hier das Problem der Computerkopplung nicht für alle Anwender gelöst werden, sondern es sollen Hinweise zur Übertragung von BASIC-Quellprogrammen und -Daten gegeben werden.
Die Programmkassette C 0171 V.24-Software enthält zwei Duplex-Schnittstellen, die vom BASIC-Interpreter des KC 85/3 über die Anweisungen LIST#2, LOAD#2, PRINT#2 und INPUT#2 (bzw. #3) angesprochen werden. Auf dem PC 1715 ist ein universelles Koppelprogramm TLCX des Büromaschinenwerkes Sömmerda als Schnittstelle zum SCPX verfügbar. Tafel 1 zeigt die Beschaltung des zur Kopplung erforderlichen Kabels. Tafel 2 gibt die einzustellenden Parameter an.
Zur Kopplung wird in den KC 85/3 entweder die Routine V24DUPL1.COM (entspricht #2) oder V24DUPL2.COM (#3) geladen. Beide Programme sind selbststartend und somit beim Start des BASIC-Interpreters sofort verfügbar. Am PC 1715 wird das Programm TLCX gestartet. Anschließend erfolgt einmalig das Einstellen der Übertragungsparameter und deren Abspeicherung. Das Aktivieren der Übertragung erfolgt mit dem Kommando 'TALK'. Durch Eingabe von '(ESC)?' können die möglichen Bedienungen abgefragt werden. Mit '(ESC)G' wird der PC auf Empfang geschaltet und mit der vereinbarten ESCAPE-Folge beendet, und die empfangenen Daten können auf Diskette abgelegt werden. Das Senden von Daten erfolgt über '(ESC)S'.
Diese Form der Übertragung ist nur für ASCII-Dateien geeignet.

Beispiel 1
Vom PC 1715 wird ein BASIC-Programm in den KC übertragen.
- Starten BASIC im KC
- Laden V24DUPL2.COM mit BLOAD
- Starten der Übertragung mit LOAD#3“A“ (beliebiger Name)
- Laden TLCX im PC
- Starten der Übertragung mit (ESC)S
- Eingabe des Namens
- Einstellen der Protokollart 'T' (TIME-WAIT, d. h. nach jeder übertragenen Zeile läßt der PC dem BASIC-Interpreter des KC eine Sekunde Zeit zum Einordnen der Zeile)
- das empfangene Programm wird auf dem Bildschirm des KC aufgelistet
- Beenden im KC mit der 'BRK'-Taste

Beispiel 2
BASIC-Programme oder Daten aus einem BASIC-Programm des KC werden als ASCII-Datei auf Diskette des PC zur weiteren Bearbeitung, z. B. mit BASI oder TP, abgelegt.
- Starten TLCX
- Starten Übertragung mit (ESC)G
- Eingabe des Namens
- Einstellen der Protokollart N (NO PROTOCOL)
- Vorbereitung am KC wie im Beispiel 1
- Ausgabe der Daten vom KC mit PRINT#3
- Abspeichern der Daten auf Diskette

Beispiel 3
Daten aus einer ASCII-Datei des PC werden zu einem BASIC-Programm in den KC übertragen
- Vorbereitung KC und PC wie im Beispiel 1
- Eingabe im BASIC-Programm durch INPUT#3 A$: PRINT#3 “Y“;
- Einstellen der Protokollart C (CHARACTER WAIT) am PC
- Eingabe des Quittungszeichens 'Y' startet die Übertragung

Modul M011: 64-KByte-RAM

Speichererweiterung für KC 85/2 und /3

Klaus-Dieter Kirves, Bernd Schenk,
Karsten Schiwon
VEB Mikroelektronik „Wilhelm Pieck“
Mühlhausen

Technische Beschreibung

Der Modul M011 ist durch folgende Funktionsbaugruppen charakterisiert:
- 64-KByte-RAM-Speicherblock mit Ansteuerung
- Modulprioritätssteuerung
- Modulsteuerung zur Zwischenspeicherung und Steuerung der Betriebsarten
- Modulkennung

Speicheradressierung und Blockstruktur

Der 64-KByte-Speicher des Moduls ist in 4 Blöcke zu je 16 KByte unterteilt. Die 4 Blöcke bilden stets einen zusammenhängenden Speicherbereich von 0000H bis FFFFH, die in den verfügbaren Adreßraum des KC-Systems eingeordnet werden. Im Bild 1 ist die Speicheraufteilung des KC 85/3 mit 64-KByte-RAM-Modul dargestellt.

Modulprioritätssteuerung

Alle Module des Kleincomputers verfügen über eine Modulprioritätssteuerung. Die Module sind deshalb in einer Kette aneinandergereiht, die durch die Signale MEI und MEO gebildet wird. Damit ist es möglich, alle aktiv geschal-

Bild 1 Speicheraufteilung Modul M011 im KC 85/3

1. Block M011		3. Block M011	4. Block M011	
Arbeits-RAM 16K	2. Block M011	BWS 16K	BASIC-Interpreter 8K	Betriebssystem 8K
0000 3FFF	4000 7FFF	8000 BFFF	C000 DFFF	E000 FFFF

Bild 2 Zuordnung der vier 16-K-Blöcke des M011 im absoluten Speicherraum des KC 85/3 in Abhängigkeit vom Steuerbyte

Steuerbyte				
CX	Block 4	Block 3	Block 2	Block 1
8X	Block 3	Block 4	Block 1	Block 2
4X	Block 2	Block 1	Block 4	Block 3
0X	Block 1	Block 2	Block 3	Block 4
.	0000 3FFF	4000 7FFF	8000 BFFF	C000 FFFF

X – zweite Stelle des Steuerbytes kk für die Definition der Betriebszustände

teten gleichartigen Module in ihrer Priorität so zu steuern, daß nur der jeweils höchstpriorisierte Modul für den Prozessor verfügbar ist. Alle anderen aktiven Module bleiben für den Prozessorzugriff gesperrt.
Aus dem Bild 1 läßt sich ablesen, daß der Speicher des Moduls M011 parallel zum Grundgerätespeicher angeordnet ist, sich also in der nächsten Prioritätsebene befindet. Lediglich in die Speicherbereiche, die durch die Grundgeräte nicht belegt werden (beim KC 85/2 und KC 85/3 der Bereich von 4000H bis 7FFFH sowie beim KC 85/2 zusätzlich die Bereiche von C000H bis DFFFH, E800H bis EFFFH und F800H bis FFFFH), ordnen sich die entsprechenden Speicherkapazitäten des M011-Moduls so ein, daß sich diese Blöcke in der höchsten Prioritätsebene befinden. Alle anderen Speicherblöcke des M011-Moduls sind durch die Prioritätskette gesperrt. Für den Prozessor sind diese Blöcke nicht unmittelbar zu erreichen.
In der Grundkonfiguration des KC-85-Systems ist der Adreßbereich von 4000H bis 7FFFH der einzige, welcher zusammenhängend einen nichtbelegten Adreßumfang von 16 KByte besitzt. Um dies auszunutzen, kann durch ein entsprechendes Steuerbyte jeweils ein Block des M011-Moduls in diesen Adreßraum eingeordnet werden. Die drei anderen Blöcke werden dabei auf neue Adreßbereiche umorientiert. In Bild 2 sind die Blockzuordnungen bzgl. des absoluten Adreßbereiches des Prozessors in Abhängigkeit vom Steuerbyte dargestellt.
Es ist zu erkennen, daß durch Variation des Steuerbytes die einzelnen Blöcke des M011-Moduls für den Prozessor im Adreßbereich von 4000H bis 7FFFH verfügbar gemacht werden können.
Darüber hinaus können durch Inaktivieren der Grundgerätespeicher weitere Blöcke des M011-Moduls in die Hauptspeicherebene des KC-Systems verlagert werden.

Modulsteuerung

Der Modul ist mit dem Befehl **SWITCH mm kk** zuzuweisen. Die Parameter mm (Modulsteckplatzadresse) und kk (Steuerbyte) sind zweistellige hexadezimale Zahlen, die aus je 8 Bit bestehen. Diese Zuweisung kann auch über das entsprechende Unterprogramm des Betriebssystems ausgeführt werden.
Der Modul übergibt bei der Abarbeitung des **SWITCH**-Befehls das Strukturbyte.
Die Betriebsarten werden programmtechnisch in der Modulsteuerung zwischengespeichert. Es können folgende Betriebsarten eingestellt werden:
– Modul aktiv bzw. inaktiv (Der Prozessor kann auf den aktiven Modul zugreifen, sofern es die Priorität ermöglicht, bzw. der Modul ist im inaktiven Zustand vom Prozessor getrennt).
– Schreibschutz (Der gesamte 64-KByte-Speicher ist schreibgeschützt und kann durch den Prozessor nur gelesen werden.)
– Blockzuweisung (Der 64-K-Speicher kann in 4 Blöcke zu je 16 K selektiert werden. Durch entsprechende Steuerbytes können die vier 16-K-Blöcke gemeinsam und gezielt in den absoluten Adreßbereich des Prozessors eingeordnet werden.)

Modulkennung

Jeder Modul erhält eine für ihn charakteristische Modulkennung, das Strukturbyte. Dieses Strukturbyte widerspiegelt den Modultyp bzw. die innere Struktur des Moduls im KC-System. Das Strukturbyte kann durch den Prozessor auch im inaktiven Zustand des Moduls gelesen werden. Dadurch kann sich der Nutzer jederzeit in einem ausgebauten System einen Überblick über die verfügbaren Module verschaffen und in Abhängigkeit davon seine Entscheidung treffen. Der 64-KByte-RAM-Modul besitzt das Strukturbyte F6H.
Nach Betätigen der RESET-Taste am KC bleibt das zugewiesene Steuerbyte im Modul erhalten, so daß eine Neuzuweisung nicht erforderlich ist.

Hinweise zur Systemkonfiguration

In den vorangegangenen Abschnitten wurden die Funktionsweise und die Modulzuweisung beschrieben. Daraus geht hervor, daß mit dem M011-Modul zwar eine nominale Vergrößerung der Speicherkapazität um 64 KByte RAM erfolgt, aber durch die Modulprioritätskette des Systems nur ein Teil der Modulspeicherkapazität direkt verfügbar ist. Durch gezielte Änderung des Modulsteuerbytes können neue Abschnitte des Modulspeichers in die Hauptspeicherebene des Systems verlagert werden.
Da der M011-Modul über einen zusammenhängenden Speicher von 0000H bis FFFFH verfügt, werden bei Zugriff des Prozessors auf den aktiven Modul alle nachfolgend aktiv geschalteten Speichermodule durch die Prioritätskette gesperrt. Ein niederpriorisierter ROM-Modul kann deshalb nie durch den Prozessor erreicht werden, wenn der M011-Modul auf einem höherpriorisierten Steckplatz aktiviert wurde.

BASIC-Interpreter und Modul M011

Durch den BASIC-Interpreter des Moduls M006 und des KC 85/3 kann ein maximaler Adreßraum von etwa 48 KByte für BASIC-Programme verwaltet werden. Um über diese Speicherkapazität verfügen zu können, sind zum Grundgerätespeicher zusätzlich 2 Module M022 EXPANDER RAM (16-KByte-RAM-Erweiterungsmodul) notwendig. Durch den Modul M011 kann diese Speicherkapazität für den Interpreter auch erreicht werden, wobei nur ein Modulsteckplatz belegt wird. Für den KC 85/3 kann das im Beispiel zusammengestellte System zur Anschauung dienen. Der BASIC-Interpreter benutzt zur Ablage der BASIC-Programme und Daten den Arbeitsspeicher des Grundgerätes und 2 Blöcke des M011-Moduls. Ein M011-Block liegt in der Hauptspeicherebene ab Adresse 4000H und der zweite Block ist parallel zum Bildwiederholspeicher angeordnet. Für den BASIC-Interpreter ist der Bildwiederholspeicher stets im inaktiven Zustand, so daß damit ein zusammenhängender Speicherbereich von 48 KByte zur Verfügung steht. Die Steuerung des Bildwiederholspeichers übernimmt eine spezielle Schnittstelle zum Betriebssystem CAOS.

Auslastung der 64 KByte Speicherkapazität

Für Maschinenprogramme besteht die Möglichkeit, diese Speicherkapazität auszunutzen. Dazu können die vier 16-K-Blöcke einzeln ab Adresse 4000H geladen werden.
Durch Variation der Steuerbytes können nun die jeweiligen 16-K-Softwarepakete aufgerufen und abgearbeitet werden.

Nutzung einer Zeichenkettenvariablen zur gleichzeitigen Abspeicherung verschiedener Werte

Die maximale Anzahl von Speichervariablen ist bei REDABAS/dBASE II mit 64 festgelegt. Dies führt manchmal zu Schwierigkeiten, wenn man innerhalb von Programmen mehr als 64 Variablen für die Zwischenspeicherung von Werten benötigt. Eine Ausweichmöglichkeit besteht natürlich im Anlegen einer speziellen Hilfsdatei. Die Nutzung einer solchen Hilfsdatei ist jedoch nicht besonders elegant und auch nicht immer möglich. Wie man mehrere Werte gleichzeitig in einer einzigen Variablen zwischenspeichert, wird mit dem Listing des kleinen Demonstrationsprogrammes aufgezeigt. Das weitgehend selbstdokumentierende Programm zählt Personen, deren Anzahl in der Datei DEMODAT (ANZAHL) mit dem jeweiligen Alter (N) abgespeichert ist.
Die dargestellte Realisierung würde die Anzahl von Personen je Jahrgang auf 99 beschränken. Dabei werden die Altersstufen 1 bis 99 berücksichtigt.
Eine generelle Beschränkung dieser Verfahrensweise ist mit der maximalen Länge von Zeichenkettenvariablen (254) bei REDABAS/dBASE II gegeben. Dabei darf das Produkt aus Anzahl und Länge der Werte nicht größer sein als die maximale Zeichenkettenlänge.
Das dargestellte kleine Demonstrationsprogramm zeigt nur eine Variante von vielen möglichen Variationen.

Gert Svenson

```
A:DEMO    .CMD                                    Bild 1

*DEMO
SET TALK OFF
*AUFBAU DER VARIABLEN MSUMM
STORE 1 TO MX
STORE "0000000000000000000000000" TO MSUMM
DO WHILE MX<8
STORE MSUMM+"0000000000000000000000000" TO MSUMM
STORE MX+1 TO MX
ENDDO
USE DEMODAT
DO WHILE .NOT. EOF
*SEQUENTIELLES ZAEHLEN
IF ANZAHL>0
STORE VAL(¤(MSUMM,1+(N-1)*2,2))+ANZAHL TO MY
*EINSCHREIBEN AN ENTSPRECHENDE STELLE DER VARIABLEN MSUMM
*IN ABHAENGIGKEIT VON N
IF N=1
STORE STR(MY,2)+¤(MSUMM,3,196) TO MSUMM
ELSE
IF N=99
STORE ¤(MSUMM,1,196)+STR(MY,2) TO MSUMN
ELSE
STORE ¤(MSUMM,1,(N-1)*2)+STR(MY,2)+¤(MSUMM,N*2+1,198-N*2);
 TO MSUMM
ENDIF
ENDIF
ENDIF
SKIP
ENDDO
USE
SET PRINT ON
? "***ALTERSSTRUKTUR***"
STORE 1 TO MN
DO WHILE MN<100
*SCHRITTWEISES AUSLESEN AUS DER VARIABLEN MSUMM
STORE ¤(MSUMM,1+(MN-1)*2,2) TO MY
IF VAL(MY)>0
? "ALTER "+STR(MN,2)+": "+MY+" PERSON(EN)"
ENDIF
STORE MN+1 TO MN
ENDDO
SET PRINT OFF
SET TALK ON
RETURN
```

Bild 1
Programmlisting DEMO

```
STRUCTURE FOR FILE:  A:DEMODAT .DBF
NUMBER OF RECORDS:   00013
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD       NAME          TYPE WIDTH   DEC
001      N               N    002
002      ANZAHL          N    002
** TOTAL **                 00005
```

Bild 2
Struktur der Datei DEMODAT

```
  DISP ALL
00001    99  23
00002    80   2
00003    12  50
00004     1  45
00005     2  30
00006     1  10
00007    99   2
00008    50  50
00009    12  24
00010    35  10
00011    33   5
00012    32   1
00013    50   5
```

Bild 3
Inhalt der Datei DEMODAT

```
***ALTERSSTRUKTUR***
ALTER  1: 55 PERSON(EN)
ALTER  2: 30 PERSON(EN)
ALTER 12: 74 PERSON(EN)
ALTER 32:  1 PERSON(EN)
ALTER 33:  5 PERSON(EN)
ALTER 35: 10 PERSON(EN)
ALTER 50: 55 PERSON(EN)
ALTER 80:  2 PERSON(EN)
ALTER 99: 25 PERSON(EN)
```

Bild 4
Ergebnisausdruck Programm DEMO

Speicherplatz sparen

Häufig sind bei bestimmten Anwendungen größere Datenlisten abzuspeichern, und wenn dies auf der Grundlage von BASIC-Feldern erfolgt, kann es bei einem KC sehr schnell recht eng werden. Handelt es sich darüber hinaus um Integerzahlen (z. B. Bildschirmkoordinaten, reale Koordinaten von Flächen oder Körpern in Graphikprogrammen), dann ist die Verwendung von BASIC-Feldern reine Speicherplatzverschwendung.

Manche Interpreter bieten für diesen Fall die Möglichkeit der Definition von Integervariablen; beim KC 85/2 (3) gibt es das leider nicht.
Unter der Voraussetzung, daß alle Daten innerhalb eines bestimmten Intervalls liegen (z. B. $0 \leqq x \leqq 255$, Bildschirmkoordinaten für ein entsprechendes Fenster), kann man durch interne Kodierung 50 Prozent und mehr an Speicherplatz einsparen.

Grundprinzip:

Vom BASIC-Speicher wird am oberen Ende ein hinreichend großer Block abgetrennt. Beginnend mit der Anfangsadresse AA wird die Liste im Stapelverfahren mit aufsteigenden Adressen abgelegt. Die Nummer eines Listenelements wird als Zeiger benutzt.

Beispiele

① Integerzahlen z ($0 \leqq z \leqq 255$)
Ablage: POKE AA+NR,Z
Zugriff: Z=PEEK (AA+NR)
② Integerzahlen z ($-32768 \leqq z \leqq +32767$)

In diesem Fall werden die Daten intern als HEX-Zahlen (2 Byte) abgelegt. Die Konvertierung und Rekonvertierung besorgt der Interpreter automatisch. Im Gegensatz zu POKE werden bei DOKE und DEEK negative Argumente akzeptiert und richtig interpretiert.
Ablage: DOKE AA+2*NR,Z
Zugriff: Z=DEEK (AA+2*NR)
③ Eine weitere Möglichkeit der internen Kodierung ist die Verschlüsselung der Daten als Zahlen zur Basis 100. Nennen wir sie „HEKTO-Zahlen". Als Kodes für die „HEKTO-Ziffern" 0 bis 99 kann man z. B. die für Steuer- und Sonderzeichen im KC 85/3 verwendeten Kodezahlen von 91 bis 190 definieren. Im PRINT wären diese Zahlen kaum lesbar; das ist aber unwichtig, da Kodierung und Dekodierung nur programmintern erfolgen.

Ablage:
Z1=INT(Z/100):Z2=(Z/100−Z1)*100
Z1=Z1+91:Z2=Z2+91
POKE AA+2*NR,Z1:POKE AA+2*NR+1,Z2
Zugriff:
Z1=PEEK (AA+2*NR):Z2=PEEK (AA+2*NR+1)
Z=100*(Z1−91)+Z2−91
Mit diesem Beispiel können Daten aus dem Intervall $0 \leqq Z \leqq 9999$ kodiert werden.
Wer sich die „HEKTO-Zahlen" ansehen möchte, der kann bei der Kodierung ergänzen:
Z$=CHR$(Z1)+CHR$(Z2):?Z$

HEKTO-Zahlen bieten im angegebenen Intervall gegenüber der HEX-Verschlüsselung sicher keine Vorteile, sie könnten aber bei sehr großen Zahlen oder in Listenverarbeitungsprogrammen interessant werden.
Die im abgetrennten Speicherbereich abgelegten Listen können als Files mittels CAOS SAVE gerettet und über CAOS LOAD (bzw. BLOAD) wieder geladen werden. Durch Angabe eines Offset-Parameters bei CAOS LOAD kann der File auf einen anderen Block geladen werden; gibt man den Parameter im Zweierkomplement an, erfolgt das Laden auf eine niedrigere Adresse als die des Quellblocks.
Die angeführten Beispiele sind als Denkanstoß gedacht, wenn es darum geht, bei einem bestimmten Programm eine hinsichtlich der Speicherökonomie vorteilhafte Datenstruktur bzw. Speicherstruktur festzulegen.

H. Bamberger

Leistungsbilanz der Stromversorgungsbaugruppe von KC 85/1

Das Netzteil der robotron-Kleincomputer ist für die Bereitstellung folgender Versorgungsspannungen und Lastströme ausgelegt:

+5 V ± 2,5 %	3,5 A
+12 V ± 10 %	450 mA
−5 V ± 10 %	100 mA
−12 V ± 10 %	150 mA

Tafel 1

Gerät bzw. Ergänzung	+5 V	+12 V	−5 V	−12 V
KC 85/1 Grundgerät	2,0 A	140 mA	120 mA	110 mA
KC 87 Grundgerät	2,0 A	80 mA	45 mA	60 mA
BASIC-Modul	0,3 A	–	–	–
RAM-Modul	0,25 A	50 mA	2 mA	–
Farb-Modul	0,5 A	–	–	–
Drucker-Modul	0,33 A	15 mA	–	15 mA
E/A-Modul	0,07 A	–	–	–
ADU-Modul	0,12 A	30 mA	–	10 mA
Programmier-Modul	0,18 A	120 mA	–	70 mA
Spracheingabe-Modul	0,11 A	20 mA	5 mA	–

Die Gesamtbelastung darf 25 W nicht überschreiten, d. h., wenn z. B. die +12 V-Ausgangsspannung nur mit 225 mA (2,7 W) belastet ist, kann die Belastung von +5 V noch bis maximal 4 A gesteigert werden.
Für den Computer (Grundgerät) und Erweiterungsmodule wurden dabei die in Tafel 1 angegebenen typischen Werte der Lastströme ermittelt.
Die Differenz zwischen den maximal zulässigen Lastströmen und den entsprechend der betriebenen Konfiguration tatsächlich entnommenen Lastströme darf am Modulsteckverbinder bzw. am Anwendersteckverbinder an den entsprechenden Kontakten (s. Bedienungsanleitung) entnommen werden.

Dr. G. Kleinmichel

Farbbildausgabe von KC 85/1 und KC 87

Die Farbbildausgabe erfolgt bei robotron-Kleincomputern im Interesse einer optimalen Bildqualität ausschließlich über Farbfernsehgeräte mit RGB-Ansteuerung. Dieser Steuereingang kann bei Farbfernsehgeräten des Fernsehgerätewerkes „Friedrich Engels" Staßfurt (ab Serie 4000) bzw. bei allen robotron-Kofferfarbfernsehgeräten nachgerüstet werden. Dabei wird durch eine Vertragswerkstatt des VEB Industrievertrieb Rundfunk und Fernsehen der *Ergänzungssatz Farbe 690 016.4,* der über das Kleincomputer-Vertriebsnetz erhältlich ist, in die geeigneten Fernseher eingebaut. Dieser Einbau erfolgt ohne Garantieverluste oder Konsequenzen auf die Fernsehempfangsqualität. In jedem Bezirk sind mindestens 2 Werkstätten für den Einbau der Ergänzungssätze geschult und autorisiert.
Computerseitig sind alle Geräte mit den Typenbezeichnungen KC 85/1.11 oder KC 87.11 für Farbwiedergabe eingerichtet. Computer mit der Typenbezeichnung KC 85/1.10 bzw. KC 87.10 müssen durch Einbau eines *Farbmoduls 690 005.1* farbtüchtig gemacht werden. Dieser Einbau wird in den Computer-Service-Werkstätten vorgenommen.
Achtung! Erfolgt die Bildausgabe eines farbtüchtigen Computers über den Antennenanschluß auf ein Schwarzweiß-Fernsehgerät, so ist zunächst keine Kursordarstellung auf dem Bildschirm vorhanden. Im Anlaufprogramm wird nämlich mit dem Erkennen des Vorhandenseins des Farbattributspeichers bei farbtüchtigen Computern in einem 64-Bit-Speicher-Register ein entsprechendes Kennbit gesetzt (Adresse EFC8H, Bit 5). Daraufhin erfolgt die Kursormarkierung durch Setzen des Blinkbits im Farbattributspeicher (siehe Programmierhandbuch S. 112) für die aktuelle Kur-

sorposition. Die Generierung des s/w-Kursors am Antennenausgang wird unterdrückt. Erst durch Rücksetzen des entsprechenden Kennbits, z. B. BASIC-Befehl POKE –4152,16 (Adresse –4152 entspricht in der BASIC-Notierung der Adresse EFC8H und der Zelleninhalt 16 (oder binär 0001 0000) kennzeichnet die Speicherbereiche E000H – EBFFH und F000H – FFFFH als ROM-Bereiche und nur den Bildwiederholspeicher EC00H – EFFFH als RAM-Bereich) wird aus dem Farbgerät scheinbar wieder eine s/w-Konfiguration und der Kursor erscheint wieder auf dem s/w-Bildschirm (siehe auch entsprechenden Hinweis, z. B. auf S. 18 des Anhangs zum Programmierhandbuch).

Dr. G. Kleinmichel

MBASIC-Programm für den quasigrafischen Zeichengeneratorentwurf

Arnd Kempe
Amt für Standardisierung, Meßwesen und Warenprüfung Berlin

Üblicherweise beziehen sich Zeichengeneratoren auf die Rastermaße 8×8 (Homecomputer) oder 8×16 (Bürocomputer, PC 1715). Die sie enthaltenden EPROMs zählen meist zur Firmware. Änderungen nimmt man wegen nicht vorhandener Editierhilfen kaum vor. Was aber, wenn für Fachtexte griechische Buchstaben benötigt werden? Oder man die fehlenden Umlaute in den Zeichensatz aufnehmen will? Oder wenn man ganz spezielle Symbole verwenden möchte (die nicht jedesmal neu definiert werden sollen)? Mit dem MBASIC-Programm SIGNGEN wird für all diese Fälle ein Werkzeug zur Verfügung gestellt (Voraussetzung: Betriebssystem CP/M).

In Stichpunkten:

– Das Programm ist auf unterschiedliche Hardwarebedingungen hinsichtlich der Videoansteuerung vorbereitet (z. B. Anzeige von weniger als 8 Spalten, um die Bildschirmkapazität zu vergrößern, freibleibende Cursorzeilen, ...).
– Das Entwerfen alphanumerischer Zeichen wird durch eine spezielle Maske unterstützt.
– Ein bereits als Hexfile vorhandener Zeichensatz (vorgegebener Name der Diskettendatei: SIGN.OLD) kann zum Vergleich in einer zweiten Maske angezeigt und in der Editiermaske verändert werden. Das ist besonders von Vorteil, wenn man nur kleine Änderungen vornehmen will.
– Mit dem P-Kommando ist eine komfortable Dokumentation des Generators ausdruckbar.
– Beim Verlassen des Programms wird der Hexcode unter SIGN.NEU auf Diskette abgelegt.

Informationen zu den einzelnen Funktionen sind in den Zeilen 100–150 des Listings nachlesbar. Genauer ansehen sollte man sich auch Zeile 200: Hier wird die spezielle Tastatur angepaßt. Wer mit der robotron K7636 arbeitet, läßt die Zeile unverändert. Anderenfalls müssen die Codes der Cursortasten und des DELETE-Keys mit einem kurzen Programm ermittelt werden:

10 PRINT ASC(INPUT$(1)):GOTO 10

Bis zum Bedienen von CONTROL C

1a

```
100 'Programm SIGNGEN zum quasigrafischen Editieren von Zeichengeneratoren
    Wahlweise 1 x 1k- EPROM mit 128   8x8- Zeichen
              2 x 1k- EPROM mit 128  16x8- Zeichen
              1 x 2k- EPROM mit 128  16x8- Zeichen
110 'Das Programm akzeptiert folgende Eingaben:
    Im Editiermodus:    "*" bzw. " "
                        Die Cursortasten        ^
                                                |
                                        <--           -->
                                                |
                                                v
                        fuer die Bewegung in der Eingabemaske
120 '                   DEL loescht die Eingabemaske
    Im Kommandomode:    N       Editieren des nachfolgenden ASCII-
                                Zeichens
                        S       Sprung zu bel. ASCII- Code
                        A       Maske fuer alphanum. Symbole
                        G       Maske fuer Grafiksymbole
                        C       Druck des gerade editierten Zeichens
130 '                   P       Druck des gesamten neuen Zeichensatzes
                        E       Exit zum CP/M
140 'Standardbezeichnungen fuer die Zeichengeneratordateien (Hexcode)
    sind:
    SIGN.OLD            zu aendernder Zeichensatz (falls nicht vorhanden,
                        bleibt die Vergleichsmaske leer)
    SIGN.NEW            neuerstellter Zeichensatz
                        Vorbelegt: SIGN.NEW=SIGN.OLD
150 'Aufruf: [LW1:]basi [LW2:]signgen
170 '*****************************************************************************
180 'Definition der Bildschirm- Steuercodes fuer CP/M
190 CLRS$=CHR$(12):CLRL$=CHR$(24):
        LEFT.$=CHR$(8):RIGHT.$=CHR$(21):
        DOWN$=CHR$(10):UP$=CHR$(26)
195 'Anpassung der speziellen Tastatur (B.B. aendern!)
    Nachfolgend fuer ROBOTRON K7636
200 LEFT.KEY$=CHR$(8):RIGHT.KEY$=CHR$(21):
    DOWN.KEY$=CHR$(5):UP.KEY$=CHR$(4):
    DEL.KEY$=CHR$(127)
205 'Definition der Funktionen
210 DEF FNCURS.POS$(X,Y)=CHR$(27)+CHR$(127+X)+CHR$(127+Y):
    DEF FNSET(BYTE,BIT)=BYTE OR 2^BIT:
    DEF FNLOOK(BYTE,BIT)=(BYTE AND 2^BIT)()0:
    DEF FNINSERT$(OLD$,BYTE$,POSITION)=LEFT$(OLD$,POSITION-1)+BYTE$
                                +RIGHT$(OLD$,LEN(OLD$)-POSITION)
220 '
250 'Eingangsdialog und Definitionen
260 PRINT CLRS$;
    TAB(14);"*** RASTERDEFINITIONEN FUER ZEICHENGENERATOR ***":
    PRINT:PRINT"I.  Eingabe globales Raster: 8*8 (1) oder 8*16 (2) ";
270 SIZE$=INPUT$(1):IF SIZE$()"1" AND SIZE$()"2" THEN 270
        ELSE PRINT SIZE$:SIZE%=VAL(SIZE$)*8
275 DIM SIGN.OLD$(SIZE%),SIGN.NEW$(SIZE%)
280 PRINT:
    PRINT"II. Angabe spezielles Raster fuer alphanumerische Symbole"
290 PRINT "     Anzahl freier Spalten am linken Rand:    ";:
        I$=INPUT$(1):IF I$("0" OR I$)"2" THEN PRINT CLRL$;:GOTO 290
                                        ELSE PRINT I$:LINKS.ALPHA%=VAL(I$)
300 PRINT "     Anzahl freier Spalten am rechten Rand:  ";:
        I$=INPUT$(1):IF I$("0" OR I$)"2" THEN PRINT CLRL$;:GOTO 300
                                        ELSE PRINT I$:RECHTS.ALPHA%=VAL(I$)
305 PRINT "     Anzahl der unten freibleibenden Zeilen: ";:
        I$=INPUT$(1):IF I$("0" OR I$)"5" THEN PRINT CLRL$;:GOTO 305
                                        ELSE PRINT I$:UNTEN.ALPHA%=VAL(I$)
310 PRINT:
    PRINT"III.Angabe nicht programmierbarer Spalten/Zeilen fuer Grafikzeichen"
```

1b

```
320 PRINT "     Anzahl derartiger Spalten: ";:
        I$=INPUT$(1):IF I$("0" OR I$)"3" THEN PRINT CLRL$;:GOTO 320
                                        ELSE PRINT I$:GRAFIK.SPALTEN%=VAL(I$)
330 PRINT "     Anzahl derartiger Zeilen:  ";:
        I$=INPUT$(1):IF I$("0" OR I$)"5" THEN PRINT CLRL$;:GOTO 330
                                        ELSE PRINT I$:GRAFIK.ZEILEN%=VAL(I$)
340 '
350 '*** Schirm zeichnen
360 PRINT CLRS$;
    TAB(22);"*** CREATING A NEW CHARACTER SET ***":
    PRINT TAB(20);"OLD SIGN";TAB(53);"NEW SIGN":
        IF SIZE%=16 THEN 370 ELSE PRINT:PRINT:PRINT:PRINT
370 LINIE$="+--------+":
    PRINT TAB(19);LINIE$;TAB(52);LINIE$:
        FOR I%=1 TO SIZE%:
                PRINT TAB(19);"|";TAB(28);"|";TAB(52);"|";TAB(61);"|":
        NEXT:
    PRINT TAB(19);LINIE$;TAB(52);LINIE$
380 IF SIZE%=8 THEN PRINT:PRINT:PRINT:PRINT
390 PRINT STRING$(79,"="):
    PRINT"(N)ext  (S)kip  (A)lphanumeric  (G)rafik  (C)opy  (P)rint  (E)xit"
400 PRINT FNCURS.POS$(9,33);"Actual Ascii";
    FNCURS.POS$(11,33);"Chr:  ' '";
    FNCURS.POS$(12,33);"Dec:";
    FNCURS.POS$(13,33);"Hex:";
500 '
510 '*** Einlesen SIGN.OLD, falls vorhanden
520 ON ERROR GOTO 2000:OLD.EXIST%=1:OPEN "I",#1,"SIGN.OLD":
    CLOSE:ON ERROR GOTO 0
530 OPEN "R",#1,"SIGN.OLD",128:FIELD 1,128 AS A$
540 FOR I%=1 TO SIZE%:
        GET 1,I%:SIGN.OLD$(I%)=A$:
    NEXT:
    CLOSE
550 ON ERROR GOTO 0
555 '
560 '*** Kommandobereitschaft herstellen
570 IF OLD.EXIST% THEN
        FOR I%=1 TO SIZE%:
                SIGN.NEW$(I%)=SIGN.OLD$(I%):
        NEXT:
              ELSE
        FOR I%=1 TO SIZE%:
                SIGN.NEW$(I%)=STRING$(128,CHR$(0)):
        NEXT
580 CODE%=0:GRAFIK%=0:OBEN%=11-(SIZE%-2)/2:GOSUB 1300
                                                'Zeichnen Grafikgrenzen
600 '
610 '*** Hauptschleife - Herstellen Belegung fuer aktuellen CODE%
625 K%=INT(CODE%/16)+1:PLACE%=(CODE%-16*INT(CODE%/16))*8
                                                'Ermitteln Feldelement
                                                 SIGN.OLD$(K%) + Position
630 FOR I%=0 TO SIZE%-1:OFFSET%=INT(I%/8)*8:PLATZ%=PLACE%+I%+1:
        FOR J%=0 TO 7:ADR%=&HF800+(OBEN%-1+I%)*80+19+J%:
                IF OLD.EXIST% THEN
                IF FNLOOK(ASC(MID$(SIGN.OLD$(OFFSET%+K%),PLATZ%,1)),7-J%)
                        THEN POKE ADR%,42
                        ELSE POKE ADR%,32
635             IF FNLOOK(ASC(MID$(SIGN.NEW$(OFFSET%+K%),PLATZ%,1)),7-J%)
                        THEN POKE ADR%+33,42
640     NEXT J%:
    NEXT I%
654 PRINT FNCURS.POS$(11,40);:IF CODE%)32 AND CODE%(127
        THEN PRINT CHR$(CODE%)
        ELSE PRINT " "
657 PRINT FNCURS.POS$(12,39);USING "###";CODE%:
```

```
    PRINT FNCURS.POS¤(13,40);:IF CODE%(16 THEN PRINT " ";
660 PRINT USING"\\";HEX¤(CODE%)
670 PRINT FNCURS.POS¤(24,1);"COMMAND) ";
    FNCURS.POS¤(OBEN%,LINKS%);:VERTIKAL%=OBEN%:HORIZONTAL%=LINKS%
700 '
710 '*** Kommandostrickleiter
720 I¤="":I¤=INPUT¤(1)
730 'Cursorbewegungen und Zeichnen
740 IF I¤=LEFT.KEY¤ THEN
       IF HORIZONTAL%-1(LINKS% THEN 720
                                ELSE PRINT LEFT.¤;:HORIZONTAL%=HORIZONTAL%-1:
                                        GOTO 720
750 IF I¤=RIGHT.KEY¤ THEN
       IF HORIZONTAL%+1)RECHTS% THEN 720
                                 ELSE PRINT RIGHT.¤;:HORIZONTAL%=HORIZONTAL%+1:
                                        GOTO 720
760 IF I¤=UP.KEY¤ THEN
       IF VERTIKAL%-1(OBEN% THEN 720
                             ELSE PRINT UP¤;:VERTIKAL%=VERTIKAL%-1:GOTO 720
770 IF I¤=DOWN.KEY¤ THEN
       IF VERTIKAL%+1)UNTEN% THEN 720
                              ELSE PRINT DOWN¤;:VERTIKAL%=VERTIKAL%+1:GOTO 720
780 IF I¤()" " THEN 790 ELSE PRINT" ";:GOSUB 1000:GOTO 720
790 IF I¤()"*" THEN 800 ELSE PRINT"*";:GOSUB 1000:GOTO 720
800 IF I¤="N" THEN GOSUB 1100:GOSUB 1200:GOSUB 1400:CODE%=CODE%+1:GOTO 600
810 IF I¤()"S" THEN 820
812    GOSUB 1100:GOSUB 1200:GOSUB 1400
814    PRINT FNCURS.POS¤(24,20);"Code(Dezimal): ";:
       INPUT;"",CODE%:IF CODE%(0 OR CODE%)127 THEN 814 ELSE PRINT CLRL¤;:GOTO 60^
820 IF I¤()"A" THEN 830 ELSE GOSUB 1100
822    'loeschen Grafikmaske, zeichnen Alphamaske
824    IF GRAFIK%=0 THEN 670 ELSE GRAFIK%=0
826    IF GRAFIK.SPALTEN%(LINKS.ALPHA% AND GRAFIK.SPALTEN%)0 THEN
            FOR I%=OBEN% TO OBEN%+SIZE%-1-GRAFIK.ZEILEN%:
                 PRINT FNCURS.POS¤(I%,52+GRAFIK.SPALTEN%);" ";:
            NEXT
827    IF GRAFIK.ZEILEN%(UNTEN.ALPHA% AND GRAFIK.ZEILEN%)0
            THEN PRINT FNCURS.POS¤(I%,53);STRING¤(8," ")
828    LINKS%=53+LINKS.ALPHA%:RECHTS%=60-RECHTS.ALPHA%:
       UNTEN%=OBEN%+SIZE%-1-UNTEN.ALPHA%
829    FOR I%=OBEN% TO UNTEN%:
            PRINT FNCURS.POS¤(I%,LINKS%-1);"|";
            FNCURS.POS¤(I%,RECHTS%+1);"|":
       NEXT:
       PRINT FNCURS.POS¤(I%,53);STRING¤(8,"-"):
       GOTO 670
830 IF I¤="G" THEN GOSUB 1100:GOSUB 1300:GRAFIK%=1:GOTO 670
840 IF I¤()"C" THEN 850 ELSE GOSUB 1100
842    LPRINT:LPRINT:
       FOR I%=0 TO SIZE%+1:
            FOR J%=51 TO 60:
                 LPRINT CHR¤(PEEK(&HF800+(OBEN%-2+I%)*80+J%));:
            NEXT J%:
            LPRINT:
       NEXT I%
844    LPRINT " ASC=";USING "###";CODE%;:LPRINT "D":
       LPRINT "    =";USING "###";VAL(HEX¤(CODE%));:LPRINT "H"
846    IF CODE%)32 AND CODE%(127 THEN LPRINT "Chr   =  ";CHR¤(CODE%)
848    GOTO 670
850 IF I¤()"P" THEN 950 ELSE GOSUB 1100:LPRINT:LPRINT
855    FOR I%=0 TO 15
857    P%=INT(I%/2)+1:R%=INT((I%+1)/2+1-P%)     'Feldelement in SIGN.NEW¤
                                                  und Rest, da je 2 Makro-
                                                  zeilen in einem Stringelement
860    LPRINT "ASC:";:IF I%(4 THEN LPRINT:GOTO 870
865    FOR J%=0 TO 7:
```

1c

```
            IF I%*8+J%(127 THEN LPRINT TAB(J%*9+11);CHR¤(I%*8+J%);
867    NEXT J%:LPRINT
870    LPRINT "HEX:";:
       FOR J%=0 TO 7:
            LPRINT TAB(J%*9+10);:IF I%*8+J%(16 THEN LPRINT " ";
872         LPRINT USING"\\";HEX¤(I%*8+J%);:
       NEXT J%:LPRINT
875    LPRINT "DEC:";:
       FOR J%=0 TO 7:
            LPRINT TAB(J%*9+9);USING"###";I%*8+J%;:
       NEXT J%:LPRINT
880    LPRINT"     +";STRING¤(71,"-");"+"
885    FOR J%=1 TO SIZE%:OFFSET%=8*INT((J%-1)/8)
                                                   'Zeilen in Makrozeile
890    LPRINT"     |";:
            FOR L%=0 TO 7:PLATZ%=R%*64+L%*8+J%:
            FOR M%=0 TO 7:
            IF FNLOOK(ASC(MID¤(SIGN.NEW¤(OFFSET%+P%),PLATZ%,1)),7-M%)
                 THEN LPRINT "*";
                 ELSE LPRINT " ";
895         NEXT M%:LPRINT "|";:
            NEXT L%:LPRINT                       'l%: Kaesten pro Zeile
                                                  m%: Zeichen pro Kasten
900    NEXT J%
930    LPRINT "     +";STRING¤(71,"-");"+":LPRINT
935    NEXT I%:GOTO 670
950 IF I¤()"E" THEN 970
955    GOSUB 1100:GOSUB 1200:GOSUB 1400
960    OPEN "R",#1,"SIGN.NEW",128:FIELD 1,128 AS A¤:
       FOR I%=1 TO SIZE%:
            LSET A¤=SIGN.NEW¤(I%):
            PUT 1,I%:
       NEXT
965    CLOSE:PRINT CLRS¤;:SYSTEM
970 IF I¤()DEL.KEY¤ THEN 670                     'Ende Kommandostrickleiter
975    GOSUB 1400
980    'kill passendes Feldelement
982    BYTE¤=CHR¤(0):
       FOR I%=1 TO 8:
            SIGN.NEW¤(K%)=FNINSERT¤(SIGN.NEW¤(K%),BYTE¤,PLACE%+I%):
            IF SIZE%=16 THEN
            SIGN.NEW¤(8+K%)=FNINSERT¤(SIGN.NEW¤(8+K%),BYTE¤,PLACE%+I%)
984    NEXT
985 GOTO 670
997 '
998 '
999 '***************** UNTERPROGRAMME ******************
1000 HORIZONTAL%=HORIZONTAL%+1:
       IF VERTIKAL%/2=VERTIKAL%\2 THEN 1010 ELSE 1020
1010 IF HORIZONTAL%)RECHTS% AND UNTEN%)VERTIKAL%
       THEN PRINT LEFT.¤+DOWN¤;:
            VERTIKAL%=VERTIKAL%+1:HORIZONTAL%=HORIZONTAL%-1
       ELSE IF HORIZONTAL%)RECHTS% AND UNTEN%=VERTIKAL%
            THEN PRINT LEFT.¤;:HORIZONTAL%=HORIZONTAL%-1
1015 RETURN
1020 IF HORIZONTAL%)LINKS%+1 THEN PRINT LEFT.¤+LEFT.¤;:
                                  HORIZONTAL%=HORIZONTAL%-2
                             ELSE IF UNTEN%)VERTIKAL% THEN PRINT DOWN¤+LEFT.¤;:
                                  VERTIKAL%=VERTIKAL%+1:
                                  HORIZONTAL%=HORIZONTAL%-1
1030 RETURN
1090 '
1100 PRINT FNCURS.POS¤(24,9);I¤;:RETURN
1190 '
1200 'UP speichern NEW SIGN im passenden Feldelement
1210 FOR I%=0 TO 7:BYTE=0:WORD=0                 'Zeilen
```

1d

```
1220    FOR J%=0 TO 7:
             BIT%=PEEK(&HF800+(OBEN%-1+I%)*80+52+J%):
             IF CHR¤(BIT%)="*" THEN BYTE=FNSET(BYTE,7-J%)
1230         IF SIZE%=16 THEN BIT%=PEEK(&HF800+(11+I%)*80+52+J%):
                  IF CHR¤(BIT%)="*" THEN WORD=FNSET(WORD,7-J%)
1240    NEXT J%:
        BYTE¤=CHR¤(BYTE):
        SIGN.NEW¤(K%)=FNINSERT¤(SIGN.NEW¤(K%),BYTE¤,PLACE%+I%+1):
        IF SIZE%=16 THEN BYTE¤=CHR¤(WORD):
             SIGN.NEW¤(8+K%)=FNINSERT¤(SIGN.NEW¤(8+K%),BYTE¤,PLACE%+I%+1)
1250 NEXT I%:
     RETURN
1290 '
1300 'UP Setzen Grafikmaske. Zunaechst loeschen Alphamaske
1305 IF GRAFIK% THEN RETURN ELSE GRAFIK%=1
1310 FOR I%=OBEN% TO OBEN%+SIZE%-1-UNTEN.ALPHA%:
        IF LINKS.ALPHA%)GRAFIK.SPALTEN%
             THEN PRINT FNCURS.POS¤(I%,52+LINKS.ALPHA%);" ";
1320    IF RECHTS.ALPHA%)0 THEN
             PRINT FNCURS.POS¤(I%,61-RECHTS.ALPHA%);" ";
1330 NEXT:
     IF UNTEN.ALPHA%)GRAFIK.ZEILEN%
        THEN PRINT FNCURS.POS¤(I%,53);STRING¤(8," ")
1340 'Setzen Maskenvariable
1350 LINKS%=53+GRAFIK.SPALTEN%:RECHTS%=60:
     UNTEN%=OBEN%+SIZE%-1-GRAFIK.ZEILEN%
1370 FOR I%=OBEN% TO UNTEN%:
        PRINT FNCURS.POS¤(I%,LINKS%-1);"|":
     NEXT:
     PRINT FNCURS.POS¤(I%,53);STRING¤(8,"-"):
     RETURN
1390 '
1400 'UP Loeschen NEW- Maske
1410 LANG%=RECHTS%-LINKS%+1
1420 FOR I%=0 TO UNTEN%-OBEN%:
        PRINT FNCURS.POS¤(OBEN%+I%,LINKS%);STRING¤(LANG%," "):
     NEXT:
     RETURN
1990 '
2000 'Errorroutine: SIGN.OLD nicht vorhanden
2010 OLD.EXIST%=0:CLOSE:RESUME 550

A)
```

1e

2

```
SIGNGEN beim Editieren

                  *** CREATING A NEW CHARACTER SET ***

            OLD SIGN                              NEW SIGN


            +--------+                           +---------+
            |   **   |                           |   **    |
            |   **   |      Actual Ascii         |   **    |
            |********|                           |  ****   |
            |  ****  |        Chr: ' '           | ******  |
            |  ****  |        Dec: 13            |* **** **|
            |  ****  |        Hex:  D            |* ****   |
            |   **   |                           |  *  *   |
            |   **   |                           |  *   *_ |
            +--------+                           +---------+


==========================================================================
(N)ext  (S)kip  (A)lphanumeric  (G)rafik  (C)opy  (P)rint  (E)xit

COMMAND)
```

Bilder 1a–e Listing des Programms SINGNEN

Bild 2 SINGNEN beim Editieren

werden die Codes der betätigten Tasten ausgegeben und sind passend in Zeile 200 einzutragen.

Und noch ein Hinweis: Da die Befehlscodes „*“ sowie „“ sämtlich auf der SHIFT-Ebene liegen, sollte man die Tastatur im UPPER CASE verriegeln.

SIGNGEN kann aber nicht die Hexdatei auf den EPROM brennen.

Ein passender EPROMMER muß über einen Treiber verfügen, der auf CP/M-Diskettendateien zugreift.

Vom ASMW ist ein derartiges System (mit einem bis 64 K-EPROMMER) nachnutzbar.

Termine

Fachtagung Elektronische Stromversorgungstechnik

WER? Fachverband Elektrotechnik in der KDT, Fachausschuß 14, Steuerungs- und Regelungstechnik
WANN? 1. Oktober 1987
WO? Berlin, Kongreßhalle Alexanderplatz
WAS?
- Schaltnetzteile und andere moderne Stromversorgungskonzeptionen in Geräten und Anlagen
- Funktion und Einsatz neuer Bauelemente
- Erläuterung perspektivischer Schaltungskonzeptionen

WIE? Teilnahmemeldungen schriftlich an: Kammer der Technik, Präsidium, Fachverband Elektrotechnik, PSF 1315, Berlin, 1086.

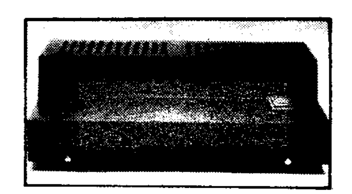
Hoppe



Börse

EPROM-Programmiersystem

Dieses System umfaßt einen Hardwareteil (s. Bild) und ein Bedienprogramm für den Wirtsrechner (z. B. A 5110/A 5120, PC 1715, PRG 700 unter UDOS).



Der Hardware-Teil (Programmiergerät PG) verfügt über autonome Stromversorgung und ist mit dem Wirtsrechner über eine V.24-Schnittstelle gekoppelt (dreipolige Verbindung).
Kernstück des PG ist ein EMR U 8820, der den seriellen Datentransfer (9600 Baud) und die Bereitstellung der jeweiligen Programmroutine im PG übernimmt.
Eine weitere Logik mit Anzeige des aktuellen EPROM-Typs (I2708 bis I27256) sorgt für eine typgerechte Signalfolge.
Das System gestattet die Löschkontrolle, das Lesen – wahlweise auf eine Zieldatei oder in den Hauptspeicher – sowie das Programmieren von einer Quelldatei. Betriebsart, EPROM-Typ, Datei-Namen, Anfangs- und Endadresse des EPROMs (bei Abweichungen vom Standard) bzw. die Bedienerführung und Prüfbit-Summe werden über die Konsole des Wirtsrechners ein- bzw. ausgegeben. Die Prozedur benötigt etwa 2,5 KByte Programmbereich und je nach EPROM-Typ bis 32 KByte im Hauptspeicher. Für die Typen I2764 bis I27256 ist eine zeitsparende Hochleistungsprogrammierung vorgesehen.
Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik, PSF 964, Karl-Marx-Stadt, 9001;
Tel. 5613339.

Dr. Magerl

PC-1715-Grundkurs

An unserer Einrichtung wurde vom CAD/CAM-Zentrum unter Leitung von Prof. Klaeger für die Schulung von Mitarbeitern in Betrieben und Institutionen ein Lehrgangsmaterial erarbeitet, das eine Einführung in die Arbeit des PC 1715 gestattet. Folgende Inhalte sind dargestellt:
1. Teil: Einführung in das Betriebssystem SCP und in die Textverarbeitung TP
2. Teil: REDABAS – Dateien, Begriffe, Kommandos
3. Teil: REDABAS – Anwendung und Beispiele
4. Teil: Tabellenkalkulationssystem KP.

Jeder dieser Teile umfaßt etwa 15 Seiten und liegt als Textdatei auf Diskette vor. Die einzelnen Dateien sind gegen eine Nachnutzungsgebühr von je 100 Mark zu beziehen und für den Nachdruck freigegeben. Interesssenten wenden sich bitte an folgende Kontaktadresse:
TU „Otto von Guericke" Magdeburg, Direktorat für Forschung, PSF 124, Magdeburg, 3010.

Dr. Springer

WORDINDEX-II

Das Textverarbeitungssystem TP, das unter dem Betriebssystem SCP läuft, läßt trotz seiner Leistungsfähigkeit eine Reihe von Wünschen offen, besonders dann, wenn große Textdokumente bearbeitet werden sollen. Hier besteht mit der Nutzung von WORDINDEX-II eine Möglichkeit, zusätzliche Serviceleistungen, wie automatisches Inhaltsverzeichnis, Stichwortverzeichnis und einheitliche Überschriften und Unterschriften usw., abzufordern. An unserer Einrichtung wurde vom CAD/CAM-Zentrum die vorliegende umfangreiche englische Programmdokumentation übersetzt und als deutsche Kurzdokumentation mit einem Umfang von 15 Seiten bearbeitet. Diese Kurzdokumentation liegt als Diskettendatei vor und kann zum Preis von 200 Mark nachgenutzt werden. Interesssenten wenden sich bitte an folgende Kontaktadresse:
TU „Otto von Guericke" Magdeburg, Direktorat für Forschung, PSF 124, Magdeburg, 3010.

Dr. Springer

Programm Diskettenverwaltung

Zur Verwaltung umfangreicher Datenbestände auf Disketten wurde ein Programm *Diskettenverwaltung* zur Nutzung unter dem Betriebssystem SCP entwickelt.
Die Eintragungen der Directory werden zusammen mit einem Diskettenbezeichner in einer Verzeichnisdatei abgespeichert. Der Diskettenbezeichner kann ebenfalls in der Directory enthalten sein.
Die Verzeichnisdatei wird im REDABAS-Format aufgebaut und ist damit variabel auswertbar.
Bei REDABAS-(dBASE-) Dateien werden der Dateiaufbau, die Dateigröße und das Änderungsdatum in die Verzeichnisdatei übernommen.
Dateien im Textformat werden so aufbereitet, verdichtet und in die Verzeichnisdatei übernommen, daß der Inhalt erkennbar wird.
VEB Rohrtechnik Delitzsch, Fr.-Ebert-Str. 30, Delitzsch, 7270;
Tel. 3636.

Dr. Wernicke

Programmsystem PROBE

PROBE ist ein auf REDABAS aufbauendes universelles Programmsystem zur bildschirmorientierten Abarbeitung komplexer Qualifizierungs- und Bearbeitungsprozesse. Mittels Implementierungsprojekt erstellt der Nutzer beliebige eigene Projekte, wofür keinerlei Programmierkenntnisse notwendig sind. Der Nutzen besteht darin, daß vorhandenes Wissen über solche Abläufe erfaßt und in eine abarbeitbare Form gebracht wird. Die Ergebnisabfrage nach jedem Arbeitsschritt führt zur Präsentation des jeweils einzig richtigen Folgeschrittes. Eine REDABAS-Einführung ist als Beispiel enthalten. Zugriff zu REDABAS-Programmen und Dateien im Ablauf ist möglich.
VEB Robotron-Elektronik, BfN, Str. d. Antifa, Zella-Mehlis, 6060, Stichwort:
PROBE (NV 190/86/03).

Schlenzig

Druckertreiber für SD 1152

Zur Verkürzung der Druckzeit am A 5120/A 5130 wurde für das Betriebssystem UDOS ein Druckertreiber entwickelt. Er dient der formatierten Ausgabe von Texten und realisiert Vor- und Rückwärtsdruck bei Optimierung des Druckweges. Bei der Initialisierung des Druckertreibers SDOPT können folgende Parameter wahlweise spezifiziert werden:
- Anzahl der Zeilen je Seite
- ohne/mit Seitennumerierung
- Nummer der ersten Seite
- Anzahl der Linefeed je Zeile
- Format A3, A4 oder A5
- ohne/mit Trennlinie zwischen den Seiten
- Kopfzeile bis 32 Zeichen
- kleiner Zeilenabstand (8 Zeilen je Zoll)
- kleiner Zeichenabstand (12 Zeichen je Zoll)
- Farbbandumschaltung bei jedem Seitenwechsel.

Weiterhin ist es möglich, in einer erweiterten Variante des Treibers eine vom Anwender beliebig wählbare Umkodierung der auszugebenden Zeichen vorzunehmen.
Die Nachnutzung umfaßt den Treiber als Maschinencode, Objektcode und Quellprogramm sowie eine Kurzdokumentation.
Interessenten wenden sich bitte unter Angabe der NV-Registriernummer 27/86 an:
VEB Rohrkombinat, Stahl- und Walzwerk Riesa, Büro für Neuerer oder Abteilung Automatisierung, Kolln. Haschlar, Dimitroffstraße 10, Riesa, 8400;
Tel. 803196

Dr. Förster

Wir suchen

– eine Lösung, um Programme unter Mikro-DOS (MS-DOS) auf dem Arbeitsplatzcomputer A 7100 abarbeiten zu können,
– folgende Software für den A 7100 unter SCP 1700 (CP/M 86): Textverarbeitungssysteme, Datenbanksysteme, Tabellenkalkulationsprogramme sowie Compiler für PASCAL (möglichst Turbo), MODULA-2, PL/1, C.

Wir bieten

die Vermittlung von entsprechender Software für das BS Mikro-DOS (MS-DOS).
VEB Glasinvest Radebeul, Abt. OeO, Freiligrathstr. 9, Radebeul, 8122;
Tel. Dresden 7903106.

Haufe

Wir suchen

Programme für SM52/11

Gesucht werden für das Rechnersystem SM52/11 unter OS-RW Programme zur Grund- und Arbeitsmittelverwaltung und ähnlicher Gebiete sowie Lösungen von Dateiauswertungen (Listengenerator) und zum Magnetband-Service.
Angebote bitte an:
VEB Komplette Chemieanlagen, ORZ, Abt. 695120, Postfach 184, Dresden, 8012;
Tel. 4863309.

Dr. Gäbler

UNIX und C

Ein Anwenderhandbuch

Reihe Technische Informatik, von L. Claßen und U. Oefler, VEB Verlag Technik Berlin 1987, 236 S., 24,– M

Das Buch versteht sich als Handbuch, das während der Benutzung des UNIX-Betriebssystems als griffbereites Nachschlagewerk gute Dienste leistet. So ist auch insbesondere die Reihenfolge des dargebotenen Stoffes danach geordnet, bestimmte Themenkreise leicht aufzufinden. Viele Informationen sind in tabellarischen Zusammenstellungen – mit meist alphabetisch geordneten Schlüsselwörtern – enthalten. Didaktische Gesichtspunkte treten dabei eher in den Hintergrund. Wer sich beispielsweise als Anfänger mit dem UNIX-Betriebssystem vertraut machen möchte, beginnt am besten mit dem Kapitel 4, vor allem mit den Abschnitten 4.3.1. bis 4.3.18.
Nach der Einleitung behandelt das Kapitel 2 die Basiskonzepte des Betriebssystems UNIX. Begriffe wie UNIX-Kern, Dateien im UNIX, Prozesse und Ein-/Ausgabe unter UNIX werden erklärt. Dazu gehört auch eine Tabelle der Systemrufe des UNIX-Kerns. Das dritte Kapitel beschäftigt sich mit der UNIX-Initialisierung. Das ist für den Betreiber des UNIX von Belang, wenn man die Anwendung des UNIX als Mehrfachzugriffsystem vor Augen hat. Die UNIX-Nutzung auf einem Personalcomputer erfordert natürlich diese Kenntnisse auch für beliebige Anwender.
Kapitel 4 beschreibt die Standardoperatorprozeduren und demonstriert zugleich einen gewissen Musterdialog, der bei jeder Anwendung in diesem oder ähnlichem Sinn abläuft.
Das folgende Kapitel (40 Seiten) bringt Kurzbeschreibungen wichtiger Kommandos. Dies wird der Nutzer gewiß dankbar entgegennehmen, ebenso die Beschreibung der Bourne-Shell der zentralen Kommandosprache des UNIX-Systems. Man kann natürlich darüber diskutieren, ob nicht die attraktivere C-Shell besser geeignet wäre, oder zumindest als alternative Möglichkeit mit erklärt worden wäre.
Das letzte Drittel des Buches (80 Seiten) ist der Programmiersprache C gewidmet. Es bringt insgesamt eine gute Einführung in diese Sprache. Die Darstellung ist mit geeigneten Beispielen durchsetzt. Am Schluß ist ein Beispiel zur Nutzung von Systemrufen und Bibliotheksfunktionen angegeben. Die Beispiele wurden auf einem UNIX-System getestet und geben damit einige Gewähr für ihre Berechtigung. Mit diesem Buch wird eine Literatur-Lücke geschlossen. Dafür gebührt den Autoren und dem Verlag Achtung und Anerkennung.

Prof. Dr. Ch. Polze

Englisch–Deutsch-Fachwortschatz

Automatisierungsanlagen mit Mikrorechnern

von R. Engel, Reihe Automatisierungstechnik, H. 13., KDT, Betriebssektion des VEB GRW Teltow, Teltow 1985, 44 S.

Mit dieser Broschüre liegt eine bemerkenswerte Leistung vor, die man einem Fachkollegen bescheinigen kann, der sowohl das Gebiet der Regelungstechnik, der Mikrorechentechnik und die anglo-amerikanische Fachsprache gleichermaßen gut beherrscht. Diese Broschüre aus der KDT-Schriftenreihe des VEB GRW Teltow erreicht einen breiten Interessentenkreis durch ihre praktische Handhabbarkeit und ihr weit gefächertes inhaltliches Spektrum.

Zum Kreise der Nutzer gehören:
– die Automatisierungstechniker, die im Bereich der Forschung/Entwicklung, Konstruktion, Projektierung, Technologie und Montage tätig sind sowie
– die in der Aus- und Weiterbildung befindlichen Hoch- und Fachschulkader technischer Fachrichtungen
– und andererseits die Sprachmittler, Dokumentalisten, Kader der wissenschaftlich-technischen Information, Redakteure und Lektoren technischer Publikationen zur Aneignung fundierter Kenntnisse einschlägiger Begriffe der englischen Fachsprache, um als Mittler zwischen fortgeschrittenem Stand der Technik und Praxis wirksam werden zu können.

Das etwa 500 Fachbegriffe enthaltende Glossar ist zweisprachig aufgebaut und beinhaltet moderne Termini der Automatisierungstechnik, die in jüngster Zeit und zukünftig durch die Mikroelektronik nachhaltig beeinflußt wird.
Die Broschüre kann über die Hauptabteilung Forschungsstrategie und -ökonomie oder über die BS der KDT des VEB GRW „Wilhelm Pieck" Teltow zum Preis von 8,– Mark bezogen werden.

L. Blackert

Turbo-PASCAL

von H.-J. Joepgen, Carl Hanser Verlag München, Wien 1985, 424 S.

Das Buch wendet sich an einen breiten Leserkreis – „vom Wissenschaftler bis zum Hauptberuf-Programmierer über Informatik-Lehrer an allgemeinbildenden Schulen bis hin zu Steckenpferd-Reitern und Schülern mit wenig Programmierer-Erfahrung ...".
Dieses Kompendium ist nicht geeignet als Ersatz für das Original-Handbuch. Nur dort findet man die vollständige Auflistung aller vordefinierten Prozeduren und Funktionen oder systematische Syntax-Diagramme von Turbo.
Schon die Hauptüberschriften zeigen, auf welche Art und Weise Joepgen seinen Lesern das Turbo-PASCAL nahebringt: Einführung – Historisches und Turbos Bedeutung für die PASCAL-Welt; System-Bedienung – Technische Einzelheiten und der Umgang mit dem PDS; Streifzug – Spazieren durch die bunte Turbo-Welt; Systematik – von der inneren Ordnung der Sprach-Elemente und Magazin – Praxis-Probleme in Einzeldarstellungen.
Beim Studium der Lektüre gab es nicht nur positive Erfahrungen. Die erst vor relativ kurzer Zeit erschienene Turbo-PASCAL-Version 3.0 ist noch nicht im erforderlichen Umfang berücksichtigt. Dies ist sehr bedauerlich, denn mit dieser Version werden viele Verbesserungen eingeführt.
Als negativ wurde die (nicht sehr sachliche) Polemik BASIC–Turbo-PASCAL empfunden. Zwar weist Turbo-PASCAL gegenüber BASIC viele Vorteile auf. Der Meinung des Autors kann aber nicht zugestimmt werden, daß bei der Schaffung „anspruchsvollere(r) Programme von einigem Umfang ... der Startvorteil des BASIC-Freundes schnell dahin(schmilzt). Während er darum kämpfen muß, in einem Gestrüpp unanschaulicher GOTO- und GOSUB-Anweisungen die Übersicht zu behalten, kann sich der PASCAL-Programmierer auf die Problem-Lösung konzentrieren." Es ist mit BASIC zwar möglich, aber keinesfalls notwendig, unübersichtliche Programme zu entwickeln.
Die vorstehenden kritischen Bemerkungen ändern nichts an der positiven Einschätzung des Titels. Man sollte jedoch beachten, daß mit diesem Buch nur dann effektiv gearbeitet werden kann, wenn mindestens das Programm Turbo-PASCAL einschließlich des Handbuches sowie ein Computer, auf dem Turbo-PASCAL lauffähig ist, zur Verfügung stehen.

K.-H. Rumpf

Einführung in die Mikrorechentechnik

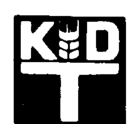
von K. Franke, 2., durchges. Auflage. VEB Verlag Technik Berlin 1986, 136 S., 64 Bilder, 13 Tafeln. 14,– M

Der Autor ordnet den Mikrorechner seinem Wesen nach in die Kategorie der Prozeßrechner ein und klärt eingangs seine Stellung in Echtzeitsystemen sowie als Instrumentarium der programmierbaren Steuerung. Danach werden die typischen Baugruppen und ihre Funktionsprinzipien vorgestellt. Dieser Abschnitt schließt mit einer kurzen Behandlung von Einchip-Mikrorechnern als hocheffektives Rechnerelement für den massenhaften Einsatzfall bei relativ kleinem Bedarf an Rechenleistung. Das letzte Kapitel befaßt sich mit der Software von Mikrorechnern. Dabei werden die Grundzüge der Assembler-Programmierung, das Wesen höherer Programmiersprachen sowie Konzepte zur Programmstrukturierung und zur Gestaltung von Echtzeitprogrammsystemen in einfacher Form erläutert.
Das Studium dieses Buches setzt beim Leser Grundkenntnisse der Rechentechnik in nur sehr geringem Umfang voraus, so daß der Charakter einer wirklich einführenden Schrift vom Grundsatz gegeben ist. Es kann festgestellt werden, daß es dem Verfasser gelungen ist, das gesteckte Ziel in vollem Umfang zu erreichen. Auffällig ist die gute pädagogische Gestaltung, die sich in der Auswahl und Reihung des Lehrstoffes, in der Klarheit und Präzision der Darlegung und in eindrucksvoll gestalteten Bildern widerspiegelt. Vielleicht wäre der Effekt noch größer gewesen, wenn der Software-Abschnitt die Assembler-Programmierung für einen konkreten Mikroprozessor komplett behandelt hätte, um damit nicht nur Überblick, sondern sogar direkte Fertigkeiten zu vermitteln. Aber das ist in Fachkreisen sehr umstritten. Die vielen Details, die bei Assemblern zu besprechen sind, gefährden natürlich den Fluß der Gesamtdarstellung.

Prof. Dr. J. Zaremba

und WAN (WAN – Wide Area Network)
• National und international wird an der Hebung des Niveaus der Softwaretechnologie für die Kommunikationstechnik gearbeitet. Dies ist verbunden mit der Nutzbarmachung standardisierter sprachlicher Hilfsmittel, z. B. ESTELLE und LOTOS (ISO-Sprachkonzepte).
• Zu den nunmehr schon traditionellen LAN-Einsatzgebieten (Rechenzentrumsbereich, Büro und Verwaltung, Prozeßsteuerung) tritt der Gesamtkomplex der künftigen hochautomatisierten Fabrik. Infolgedessen nahmen die Diskussionen des MAP- und des TOP-Konzepts relativ breiten Raum ein (MAP – Manufacturing Automation Protocol; TOP – Technical Office Protocol).
• International vollzieht sich die immer stärkere Realisierung von LAN-Funktionen durch Hardware, z. B. durch spezialisierte Schaltkreise. Auf dem Seminar wurden entsprechende experimentelle Erfahrungen aus der DDR mitgeteilt. Daneben standen zur Diskussion das Verhältnis der sogenannten Mittelklasse-LAN (bis 1 MBit/s) zu den Hochgeschwindigkeits-LAN (10 MBit/s und darüber) sowie die verstärkte Nutzung der Lichtwellenleitertechnik.
Die Seminarvorträge befinden sich im Druck und können über das Weiterbildungszentrum Informatik an der TU Dresden angefordert werden.
Für 1988 ist ein weiteres Problemseminar zum gleichen Themenkreis vorgesehen.

Prof. Dr. H. Löffler



KDT-Kolloquium „Computer und Gesellschaft '86"

Am 11. Dezember 1986 fand in Suhl das erste KDT-Kolloquium zum Rahmenthema „Computer und Gesellschaft" der wissenschaftlichen Sektion Computer- und Mikroprozessortechnik im Fachverband Elektrotechnik statt. In der Veranstaltung ging es um die Vermittlung von Informationen zu diesem komplexen Gebiet, so daß nur Plenarvorträge gehalten wurden.
Die Referenten Prof. *Posthoff* (Technische Universität Karl-Marx-Stadt), Prof. *Fuchs-Kittowski* (Humboldt-Universität zu Berlin), Prof. *Wernstedt* (Technische Hochschule Ilmenau), Prof. *M. Roth* (Technische Hochschule Ilmenau), Dr. *Groß* (Humboldt-Universität zu Berlin), Dr. *F. Roth* (Technische Hochschule Ilmenau), Dr. *Böttger* (VEB Kombinat Robotron) und Dr. *Mackrodt* (Robotron-Elektronik Dresden) sprachen zu den Themen
– Der intelligente Automat in der menschlichen Gesellschaft
– Beratungssysteme im praktischen Einsatz in der DDR
– Wissensverarbeitung – Aufgaben, Merkmale, Anwendungen
– Der Computer in der flexibel automatisierten Fabrik
– Ziele und Wege zur 5. Rechnergeneration
– Gesellschaftliche und soziale Wirkungen der modernen Informations- und Kommunikationstechnologie
– Unterstützung von Programmiersprachen in Systemen der künstlichen Intelligenz
– PROLOG – Modellierung ist gleich Programmierung.
Die 150 Teilnehmer bestätigen das Interesse an der Veranstaltung, jedoch erwies sich die fehlende Diskussionszeit als wesentlicher Mangel, so daß künftig ein zweiter Tag zur Erörterung von Fragen vorgesehen wird. Dieses Kolloquium soll im Abstand von zwei Jahren regelmäßig in Suhl stattfinden. Als Teilnehmer sind vorwiegend Ingenieure angesprochen.

Prof. Dr. M. Roth

GIDDR – Jahrestagung '86 der Fachsektion 5

(Anwendungen)

Am 21. Januar 87 fand die Jahrestagung 1986 der Fachsektion (FS) 5 (Anwendungen) Im ZKI der AdW in Berlin statt.
Ziel der Tagung war das Abstekken eines Rahmens für die Arbeit der FS in den einzelnen Arbeitsgruppen und das Vorstellen der Arbeitsgruppenthemen. Im Rahmen einer Tagesveranstaltung wurden dazu Vorträge angeboten, die sehr viele Ansatzpunkte für Diskussionen in der weiteren Arbeit aufzeigen.
Prof. Kempe (ZKI) als Vorsitzender der Fachsektion gab eine kurze Übersicht zur bisherigen Arbeit der FS und stellte einige grundlegende Anforderungen für die Anwendungen der Informatik dar. So ist z. B. durch sorgfältige algorithmische Aufbereitung ein effektiverer Einsatz der Rechentechnik nötig, um die neuen technischen Möglichkeiten mit den hohen Anforderungen in Übereinstimmung zu bringen. Mit dem Aufbau der 5 Arbeitsgruppen, deren Zielstellungen von Prof. Kempe kurz charakterisiert wurden, werden besondere Schwerpunkte der Informatikanwendung in der Volkswirtschaft behandelt, die in engem Zusammenwirken mit weiteren Gremien auf dem Gebiet der Informatik (z. B. WGMA der KDT) arbeiten. Die Vorträge am Vormittag waren als Übersichtsvorträge gedacht, um auf besondere Aufgabengebiete der Informatik hinzuweisen und spezielle Probleme der Forschungsarbeit herauszuarbeiten.
Prof. Koziolek (Zentralinstitut für sozialistische Wirtschaftsführung beim ZK der SED) konnte eindrucksvoll die Einbettung der Informatikprobleme in die gesamtgesellschaftliche Entwicklung aufzeigen. Ausgangspunkt der Überlegungen war die besondere Bedeutung der Schlüsseltechnologien, wie sie auf dem XI. Parteitag der SED dargestellt wurden. Wirtschaft und Gesellschaft sind als Ganzes aufzufassen. Die Informatik spielt dabei eine wichtige Rolle, wobei der Querschnittscharakter durch Verflechtung von verschiedenen Bereichen eine Produktivitätswirkung bedeutet. Hohes Wachstumstempo, Beachtung der Dynamik, der Produktivkraftentwicklung und der Auswirkung der wissenschaftlich-technischen Revolution stellen hohe Anforderungen an die Entwicklung der sozialistischen Betriebswirtschaft. Prof. Koziolek zeigte an Beispielen die bisherige Entwicklung auf diesem Gebiet (z. B. AUTEVO, AUTOTECH) auf und stellte die Mitwirkung des Zentralinstituts für sozialistische Wirtschaftsführung dabei dar. Bei der Arbeit aufgetretene Probleme (Fehlen von erforderlicher Hardware und Programmiersystemen) wurden erörtert. Für die weitere Arbeit des Institutes wurden die wesentlichen Schwerpunkte dargestellt (Übergang von Insellösungen zu einem Gesamtkonzept der komplexen Automatisierung/Flexibilität, Materialwirtschaft/Normung und Standardisierung, Paßfähigkeit von Hardware und Software/Rechnergestützte Qualitätssicherung im gesamten Produktionsprozeß). Der Vortrag machte deutlich, welche Bedeutung die sozialistische Betriebswirtschaft der Anwendung der Informatik beimißt und was in der weiteren Entwicklung erwartet wird.
Prof. Töpfer (TU Dresden) stellte anschließend in einem sorgfältig aufgebauten Übersichtsvortrag die besonderen Probleme der Produktionsautomatisierung dar. Einleitend erfolgte eine Begriffsbestimmung der Produktionsautomatisierung als Gesamtheit der Prozeßautomatisierung (stetige Prozesse) und der Fertigungsautomatisierung (diskrete Prozesse) und die Einordnung der Automatisierungseinrichtung als Hilfsmittel zwischen Mensch und Prozeß. Für jede Automatisierungsaufgabe sind unterschiedliche Ziele maßgebend. Als Grundziele wurden dabei die Steuerung des Produktionsablaufes, die Erhöhung der ökonomischen Ergebnisse, verbesserte Zuverlässigkeit und verbesserte Arbeits- und Lebensbedingungen genannt.
Dr. Müller (Kombinat Robotron) konnte in einem informativen Vortrag direkt auf die beiden vorangegangenen Vorträge aufbauen und zeigen, wie mit der Entwicklung der Rechentechnik versucht wird, den gestellten Anforderungen gerecht zu werden. Es wurde die besondere Bedeutung der Entwicklung neuer Software mit Softwareentwicklungssystemen hervorgehoben (Software-Engineering) und der Einfluß der Ressourcen (Hardware/Betriebssystem) darauf. Übersichtlich wurden die verschiedenen Leistungsklassen zusammengestellt und die wesentlichen Parameter erläutert. Interessant für viele Teilnehmer wird dabei besonders die weitere Entwicklung dieser wichtigen Voraussetzungen gewesen sein.
Die Notwendigkeit von Standardsoftware für breite Anwendungsklassen wurde erkannt, und daß sie in sehr kurzer Zeit zur Verfügung stehen muß.
Insgesamt gab die Tagung eine gute Übersicht über anstehende Probleme. Diskussionen waren aus Zeitgründen nur sehr eingeschränkt möglich. Weitere Veranstaltungen der FS werden von den Informationen aus den Vorträgen ausgehen können. Ausgewählte Beispiele sollen in den GI-Mitteilungen veröffentlicht werden.

Dr. J. Hübener

Rund 9000 Aussteller aller Kontinente boten in 41 Branchen auf der diesjährigen Leipziger Frühjahrsmesse vom 15. bis 21. März ihre Exponate an. Dabei offerierten mehr Aussteller als im Vorjahr Computertechnik. Unser Bericht soll einen Überblick über die in Leipzig ausgestellten Erzeugnisse dieser wichtigen Branche vermitteln. Die Berichterstattung ist untergliedert in Bauelemente, Computertechnik, periphere Geräte und lokale Netze. Informationen über die beiden zuletzt genannten Gebiete werden in MP 6/1987 veröffentlicht. Einige Neuheiten hat MP bereits teilweise recht umfangreich vorgestellt – z. B. Grafikdisplaycontroller U 82720 (MP 4/87), KC 87 (MP 1/87), P 8000 (MP 3/87), Fernschreiber F 2000 (MP 2/87) –, so daß hier nicht näher darauf eingegangen wird. Weitere ausführliche Beschreibungen von Neuentwicklungen gibt es in diesem und in den folgenden Heften, beispielsweise vom schnellen 8-Bit-D/A-Wandler C560D in dieser Ausgabe.



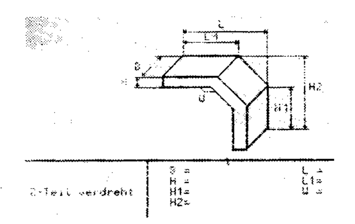
Leipziger Frühjahrsmesse 1987

Bauelemente

Wir wollen uns im Bericht auf die wesentlichen Bauelemente der Computertechnik aus den Kombinaten Mikroelektronik und Carl Zeiss JENA beschränken.
Mit dem CMOS-Gate-Array-System **U 5200** wird der Anwender in die Lage versetzt, am Entwurf seines spezifischen Schaltkreises aktiv mitzuwirken. Umfangreiche Entwicklungssoftware und ein Katalog wählbarer logischer Grundelemente erlauben eine Umsetzung der Anwenderaufgabenstellung in das Gate Array (siehe MP 1/87). Während die Grundstruktur eines Gate Array für alle Schaltkreise gleich ist (Master U5201), können drei kundenspezifische Ebenen (davon zwei Verdrahtungsebenen) modifiziert werden.
Neben der bisher gängigen Verkappung des IS in ein 64poliges Chip-Carrier-Gehäuse wurde zur LFM das PC-Sortiment (Plast-Chip-Carrier) gezeigt.
Ein weiteres Semikunden-Schaltkreissystem – auf Standardzellenbasis und damit für höhere Stückzahlen gedacht – ist das **U-1500.** Der Entwerfer hat hier die Möglichkeit, nach dem Baukastenprinzip aus einem Katalog bestimmte logische Elemente bereits im Layout festzulegen, womit Schaltungen mit einem jeweils kundenspezifischen Layout entstehen. Die Schaltkreise werden vorwiegend als SMD (oberflächenmontierbare Bauelemente) geliefert.
Dem internationalen Trend folgend, stellt die Bauelementeindustrie der DDR zunehmend Schaltungen als SMD (Surface Mounted Devices) zur Verfügung, womit sich bedeutende Einsparungen an Fläche, Volumen und Leiterplattenmaterial, eine Erhöhung der Zuverlässigkeit und Senkungen der Produktionskosten erzielen lassen. Das Sortiment umfaßt derzeit z. B. etwa 10 Typen der V-4000-Serie, 10 Typen von Einchipmikrorechnervarianten, zahlreiche industrieelle und Konsumgüterschaltkreise sowie Transistoren.
Der Einchipmikrorechner **U 8047 P** ist ein maskenprogrammierbarer Schaltkreis in CMOS-Technologie und als SMD im PCC-64-Gehäuse lieferbar. Die Programmierung erfolgt kundenspezifisch. Der Schaltkreis hat 4 Bit Verarbeitungsbreite, einen 798 × 16-Bit-ROM und einen extern erweiterbaren RAM von 64 × 4 Bit. Der U 8047 P ist für umfangreiche Aufgaben der Steuerung und Regelung sowie der Meßwerterfassung und -verarbeitung vor allem in batteriebetriebenen Geräten vorgesehen. Mit minimalem Aufwand können daher z. B. Systeme mit LCD-Anzeige und Tastatur realisiert werden.
Mit dem **U 8246 P** steht ein RAM-Schaltkreis in CMOS-Technologie zur Erweiterung des U-8047-P-RAM um 256 × 4 Bit zur Verfügung. Wie dieser wird der U 8246 P als SMD gefertigt.
Der **U 8611 DC 08** ist ein maskenprogrammierter Einchipmikrorechner mit 8 Bit Verarbeitungsbreite und einem internen ROM von 4 KByte. Die Bondversion UL 8611 DC 08 ermöglicht den Anschluß einer Batterie-Stützspannung (Datenerhalt bei power-down).
Die RAM-Kapazität beträgt 128 Byte; externe Speicher können bis zu 120 KByte adressiert werden.
Ein hochintegrierter statischer Schreib-Lese-Speicher mit wahlfreiem Zugriff (SRAM) ist der **U 6516 D** in CMOS-Technologie. Er hat eine Kapazität von 16384 Bit (2 K × 8 Bit) und wird in 3 Typvarianten gefertigt; ein zusätzlicher Bastlertyp ist in Vorbereitung. Lieferung erfolgt im 24poligen DIL-Plastgehäuse. Das U-6516-D-Sortiment vereint geringe Zugriffszeiten (150 ns bzw. 250 ns) mit geringen Ruheströmen (50 μA bzw. 5 μA) und Schlafströmen (20 μA bzw. 3 μA) und bietet besonders für die Computertechnik günstige Parameter wie Byte-Organisation, 2 Enable-Signale, Pinkompatibilität zum EPROM U 2716 C sowie TTL-Kompatibilität.
Für die Ansteuerung von bis zu 4 Normal- oder Minidiskettenlaufwerken vorgesehen ist der Peripherie-Controller-Schaltkreis **U 8272.** Er gewährleistet die Aufzeichnung von Daten wahlweise mit einfacher oder doppelter Dichte (z. B. IBM-Doppelformat, System 34, MFM).
Der Grundtyp U 8272 D08 hat eine Taktfrequenz von 8 MHz, der U 8272 D04 von 4 MHz.
Besonderheiten des U 8272 sind Mehrfachsektortransporte sowohl beim Lesen als auch beim Schreiben mit einfachem Befehl oder die interne Adreßmarkensuchschaltung.
Es besteht Pinkompatibilität zum μPD 765 und I8272.
Der **U 804 D** ist eine Schaltung in n-Kanal-Silicon-Gate-Technik und enthält einen mikrocomputergesteuerten Analogwertspeicher und D/A-Wandler für 6 Analogfunktionen mit je 6 Bit Auflösung. Über eine asynchrone serielle Schnittstelle, den CBUS, können bis zu 4 parallel betriebene Schaltkreise adressiert und gesteuert werden.

Computertechnik

Als Nachfolgemodell des PC 1715 stellte Robotron den **PC 1715W** vor. Äußerlich unverändert, verfügt er jedoch über einige wesentliche Verbesserungen. Die 8-Bit-CPU UA 880 hat gegenüber der UB 880 (PC 1715) einen schnelleren „Herzschlag" (4 MHz im Vergleich zu 2,5 MHz), was der Verarbeitungsgeschwindigkeit zugute kommt. Außerdem beträgt die RAM-Kapazität beim weiterentwickelten PC 256 KByte. Eine weitere wichtige Neuerung ist der ladbare Zeichengenerator mit einem Satz von 2 × 127 Zeichen. Als Ergänzung zum Beitrag „Semigrafik für PC 1715" in MP 1/87, S. 19 bis 21, soll hier erwähnt werden, daß der ladbare Zeichengenerator es ermöglicht, die beschriebene grafische Software ohne Hardwareveränderungen zu nutzen. Inzwischen wurde die Software weiterentwickelt. Für das Zeichnen von horizontalen und vertikalen Linien stehen in GEDIT jetzt zusätzlich 11 Sonderzeichen in zwei Strichstärken zur Verfügung. Der A3-Editor (GEDIT+) kann Bilder mit maximal 72 Zeilen und beliebig vielen Spalten bearbeiten. Beim Drucken ist das Überlagern von 3 Bildern möglich. Die Abbildung zeigt ein Anwendungsbeispiel.



Als Betriebssystem für den PC 1715W werden SCP 3.0, als Programmiersprachen BASIC, PASCAL und FORTRAN angeboten. Für PC 1715 (W)-Anwender ist sicher von Interesse, daß in diesem Jahr im Verlag Die Wirtschaft folgende Bücher zum PC 1715 herausgegeben werden:

Semigrafik für PC 1715

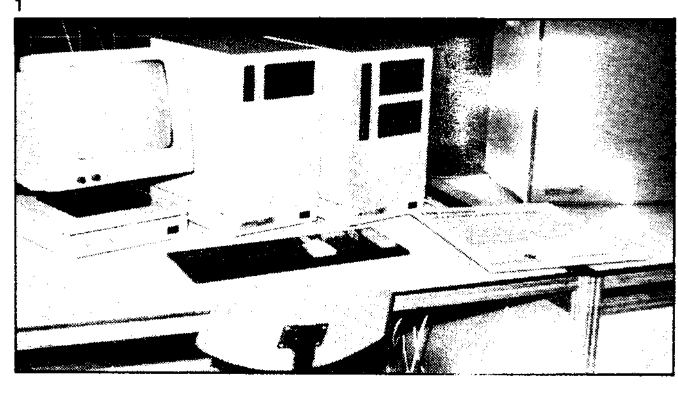
In der Technischen Hochschule Leipzig wurde im Rahmen einer Kleinserienfertigung die Möglichkeit geschaffen, die entsprechende Hardware zur Nachrüstung für den PC 1715 zur Verfügung zu stellen.
Interessenten wenden sich an die TH Leipzig, Sektion Automatisierungsanlagen, Koll. Koch, Karl-Liebknecht-Str. 132, Leipzig, 7030. Zum Lieferumfang gehören Anwenderhandbuch, Dokumentation für Anwendungsprogrammierer und Leiterplatte GM 1 (GEDIT und GEDIT+).

Der Personalcomputer **1715**, Datenbanken mit Personalcomputern, Tabellenkalkulation mit Personalcomputern, Betriebssystem SCP für Personalcomputer und Textverarbeitung mit Personalcomputern.
PC-1715-Anwendersoftware zu den verschiedensten Gebieten ergänzten die Robotron-Offerte. Dazu waren auf einem großen Teil der Ausstellungsfläche PC 1715 aufgestellt, an denen sich der Besucher jeweils über die entsprechende Lösung informieren konnte.
Für vielfältige CAD-Aufgaben einsetzbar ist der 16-Bit-Arbeitsplatzcomputer **A 7100** vom VEB Kombinat Robotron. Entsprechende grafische Peripherie ist anschließbar – auf Peripheriegeräte wird ausführlicher im 2. Teil des Berichts von der Leipziger Frühjahrsmesse 1987 in **MP 6/87** eingegangen. Jüngstes Kind der A 7100-Familie, die als CPU den UdSSR-Schaltkreis K 1810 WM 86 (kompatibel zu 8086) hat, ist der A 7100.50 mit Arithmetikprozessor, der dem Gerät mehr Geschwindigkeit verleiht. Durch erfolgte Hardwareveränderungen können auf dem AC jetzt auch MS-DOS-kompatible Betriebssysteme installiert werden. Das Gerät wird für den Einsatz eines 5,25''-Festplattenspeichers mit 10 MByte Kapazität vorbereitet.
Eine Erweiterung des P-8000-Systems aus dem KEAW ist der Industriecomputer **P 8100** (Bild 1; im Foto rechts). Er besteht aus dem Arbeitsplatzrechner P8000 (in der einfachen Variante nur mit U880/UDOS, als leistungsfähiges Leitgerät mit U 880 und U 8001 sowie WEGA als BS) und einer konstruktiv getrennten Einheit, dem Prozeßkoppelteil mit Ein-/Ausgabebaugruppen und dem Echtzeitbetriebssystem als Anbindung an den technologischen Prozeß für unterschiedlichste Anwendungen. Das P8000 dient neben der Bedienung des Gesamtsystems und der Ansteuerung weiterer Geräte über Interfaces V.24 und IFSS als leistungsfähiger Arbeitsplatzrechner. Hier erfolgen die Aufbereitung und Archivierung von Prozeß- und anderen Daten (in 5,25-Zoll-Winchesterlaufwerken größer 10 MByte) sowie die Programmentwicklung oder die Bearbeitung übergeordneter Strategien für den Echtzeitteil. Es ist Multiuserbetrieb mit max. 3 Terminals möglich. Der Prozeßkoppelteil ist ein EGS-III-Standardgehäuse mit Stromversorgungs- und Elektronik-Kassetten in 19-Zoll-Technik (EGS IV). Folgende Hardware-Moduln ermöglichen eine Vielfalt an Konfigurationsvarianten:

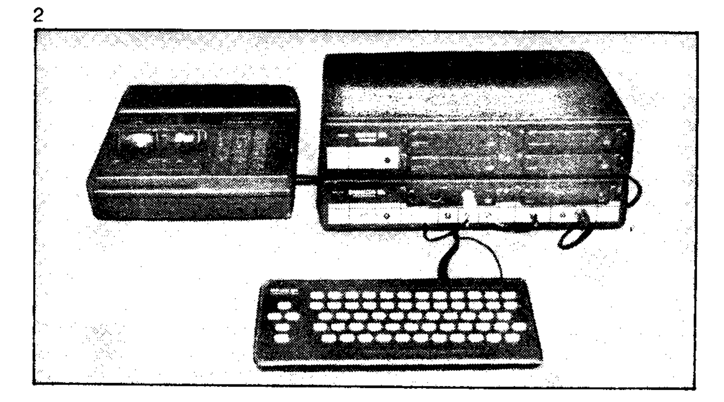
- ZVE 16-1: CPU U 8001 und MMU 8010
- ZVE 16-2: CPU U 8002
- gemischter Speichermodul SPE 1: 16-KByte-EPROM/16-KByte-CMOS-RAM
- DRAM bis 1 μByte (bei ZVE 16-1)
- Analogeingabemodul **AEM1**: speziell für sich schnell ändernde Signale, sample-and-hold-Glied, 8 Kanäle, Abtastrate 10000 Meßwerte/s
- Digitaleingabe DES 1: Polling- oder Interruptbetrieb, 16-Bit-Prozeßsignale, 24-V-Geberspannung
- Digitalausgabe DAS 1: 24-V-Pegel, Informationsbreite 15 Bit Prozeßsignale, 1 Bit Rücksetzen.
- Interfacemodul SIE 1 (Intelligenter IFSS-Modul mit U 880, Übertragungsrate 9,6 KBaud, asynchron, 5000 m max. Entfernung, 4 Kanäle bidirektional).

Für den Prozeßkoppelteil wird das Echtzeitbetriebssystem IRTS (s. MP 1/87) verwendet, als höhere Programmiersprache ist ICL verfügbar. Die Kopplung des P 8000 mit der Echtzeitgrundeinheit erfolgt über eine IFSS-Schnittstelle. In Entwicklung befindet sich ein schnelles serielles Interface IFLS (ZI, 500 KBaud).
Über den KC 85/3 vom VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen und einige Erweiterungsmodule wurde in den bisher erschienenen Ausgaben bereits ausführlich berichtet. MP wird auch weiterhin die Neuerungen vorstellen. Der KC 85/3 (bzw. /2) verfügt z. Z. schon über ein beachtliches Sortiment an Zusatzmodulen. Nachteilig war bisher, daß in das Grundgerät nur maximal zwei solcher Module gesteckt werden konnten. Durch Nutzung des **Busdriver-Aufsatzes D002** (Bild 2) sind vier Module zusätzlich nutzbar. In den Aufsatz können beliebige Module aus dem Sortiment gesteckt werden. Beim Einsatz von Speichererweiterungsmodulen passend zum KC-System (M011 64 K RAM) kann in einem Aufsatz D002 eine Speichererweiterung von 256 KByte erreicht werden. Die Aufrüstung im Turm ist über weitere Aufsätze möglich.
Zum Abschluß seien noch einige technische Parameter genannt:
Bauform: wie das Grundgerät D001
Abmessungen: 385 × 270 × 77 mm^3
Masse: etwa 4 kg
Leistungsaufnahme: etwa 15 W (220 V).
Zum Lieferumfang gehören der Aufsatz D002, Verbindungsstekker und Handbuch.
Das sowjetische Außenhandelsunternehmen ELORG zeigte eine sehr umfangreiche Exposition zur Computertechnik. So gehörten in BASIC programmierbare Taschenrechner, Einplatinen-Mikrorechner, Heimcomputer, Schulmikrorechner und leistungsfähige Grafikarbeitsplätze zum Angebot.
Als Weiterentwicklung zu den Einplatinenmikrorechnern aus der Reihe **Elektronika MC 1201** wurden die **Modelle 03** und **04** vorgestellt. Vorzüge gegenüber den Ausführungen 01 und 02 sind u. a. die Vergrößerung des Arbeitsspeichers auf 256 KByte bzw. 1 MByte und des Adreßraumes auf 1 bzw. 4 MByte. Elektronika MC 1201.03 und 1201.04 bilden die Basis für die Grafikarbeitsplätze DWK3 und DWK4.
DWK4 – es wird auch die Bezeichnung Elektronika MC 0507.10 verwendet – besitzt ein grafisches Farbdisplay (Bild 3) mit einer wahlweisen Auflösung von 400 (horizontal) × 288 (vertikal) oder 800 × 288 Punkten. 64 Farben sind möglich, dabei maximal 16 gleichzeitig. Zwei Floppy-Einheiten mit je 880 KByte und eine Festplatte mit 5 MByte Speicherkapazität sind als Externspeicher in das Gerät integriert. Der 16-Bit-Mikroprozessor K1801 WM3 soll eine Verarbeitungsgeschwindigkeit von nahezu einem Megaflops ermöglichen. DWK4 war konfiguriert mit dem Drucker EC 7189 aus der VR Polen und dem Plotter Elektronika MC 6501 mit einer Zeichengeschwindigkeit von 30 cm/s und der Positioniergenauigkeit von 0,02 mm. Vier Farben können über Programm angewählt werden.
Als erster sowjetischer Heimcomputer wurde der **Elektronika BK 0010** präsentiert. Anschließbar sind Fernsehgeräte (über Videoeingang) und Kassettenmagnetbandgerät.
Der 16-Bit-Mikroprozessor (K1801-Reihe) soll eine Verarbeitungsgeschwindigkeit von 0,3 Megaflops gewährleisten.
Als Speichergrößen sind 16 KByte Anwender-RAM, 16 KByte Bildschirm-RAM und 32 KByte ROM angegeben.
Weiterhin war ein Schulmikrocomputer zu sehen. Kurz einige Daten: Prozessor K1801 WM2, 96 KByte Anwender-RAM, 32 KByte ROM, Farbfernsehgerät anschließbar, 8 aus 32 Farben gleichzeitig darstellbar.
Bild 4 zeigt einen Lehrerarbeitsplatz, an den einige Schulmikrocomputer bzw. Heimcomputer angeschlossen sind.
Das tschechoslowakische Außenhandelsunternehmen KOVO war zur diesjährigen Frühjahrsmesse mit einer vergrößerten Ausstellungsfläche vertreten. Bereits in MP 1/87, 2. US wurde

1



2



der Rechner **CM52/12** etwas ausführlicher vorgestellt. Die in Leipzig ausgestellte Anlage hatte eine HS-Kapazität von 8 MByte. Jeweils 1 MByte sind auf einer Platine (Bezeichnung CM3211) angeordnet (Bild 5).

Das ungarische Unternehmen VIDEOTON stellte u. a. das **Mikrorechnersystem VT20/IV/M** (Bild 6) aus. VT20/IV/M ist eine Weiterentwicklung des in der DDR gut bekannten VT20A. Das System kann im 8- oder im 16-Bit-Modus arbeiten. Das ermöglichen die beiden eingebauten 8- bzw. 16-Bit-Mikroprozessoren.
Damit können unter dem Betriebssystem UPM (CP/M-kompatibel) die für den VT20A geschriebenen 8-Bit-Programme abgearbeitet werden. UDOS (MS-DOS-kompatibel) ist das Betriebssystem für den 16-Bit-Modus. Zur Grundausführung gehören 512 KByte RAM, ein 14-Zoll-Bildschirm mit 25 × 80 Zeichen bzw. einer Auflösung von 640 × 300 Punkten. Über asynchrone Interfaces sind 4 Datenstationen anschließbar. Als Systemdrucker dient der VT 23000 mit 300 Zeilen/Minute, als Hardcopy-Drucker werden die Geräte VT 21200 bzw. VT 21400 verwendet.
Bei der Ausführung mit 4 Datenstationen wird VT20/IV/M mit einer Minifloppyeinheit (1 MByte) und einem Winchester-Plattenspeicher (10 oder 20 MByte) konfiguriert, oder Datenstationen mit zwei Minifloppyeinheiten.
Der **SUPERMICRO FAMA** (Bild 7) ist eine Entwicklung des Instituts für Computer und Automatisierung der Ungarischen Akademie der Wissenschaften (MTA SZTAKI). Der Master-Control-Prozessor besteht aus einer Z8001-CPU und drei MMU Z8010. 2 MByte DRAM und 160 KByte EPROM bilden die Hauptspeicherkapazität. SUPERMICRO FAMA hat 10 serielle Schnittstellen RS-232C, zwei parallele und Centronics-Interface für Drucker, Plotter usw. Zur Standardausstattung gehören außerdem eine 5¼"-Floppy (DSDD) – DS = Double Side (doppelseitig), DD = Double Density (doppelte Dichte) – und ein Winchester-Laufwerk.

Als neue Konfigurationsvariante zeigte Mera Elzab (VR Polen) den **PC Compan 8** (Bild 8) mit 22-MByte-Winchester-Disk im Beistellgefäß. Ein weiteres Beistellgefäß enthält 2 Minifloppy-Laufwerke mit je 720 KByte. Hiervon soll es künftig auch kombinierte Minifloppy-/Winchester-Laufwerkseinheiten in halber Bauhöhe geben.
Weitere Daten des Compan 8:
– 8-Bit-CPU 8080A (MCY 7880, K 580)
– 320 KByte RAM, mit RAM-Disk 256 KByte (oder 512 KByte RAM, mit RAM-Disk 448 KByte)
– CP/M2.2 als Betriebssystem; im Verbund mit bis zu 3 Terminals wird MP/MII verwendet.

Aus dem Kombinat für Mikroprozessortechnik Pravez (VR Bulgarien) waren die Personalcomputer Pravez 8M, Pravez 16 und erstmals Pravez 16 N zu sehen. Der **Pravez 16 N** (Bild 9) wurde in der Ausführung mit 2 Diskettenlaufwerken in der Bildschirmeinheit gezeigt. Möglich sind auch Varianten ohne oder mit einem Floppy-Laufwerk bzw. mit einem Floppy- und einem Winchesterlaufwerk (10 MByte).
Der Pravez 16 N besitzt den Prozessor 8088 und den Koprozessor 8087, ist also IBM-PC-kompatibel. Bis zu 640 KByte RAM; Monochrom-Bildschirm mit 720 × 348 Punkten.
Eine **CAD-Arbeitsstation** (Bild 10) wurde von ISOT (VR Bulgarien) gezeigt. Sie ermöglicht als Offline-Grafikstation das Erfassen und Vorverarbeiten von grafischen Daten. Die dabei erstellten 1/2-Zoll-Magnetbänder können dann an einem leistungsstarken Großrechner weiterverwendet werden. Basis ist das Terminal CM 1604.M1, ergänzt um Digitalisiergerät ISOT 6411C, Plotter ISOT 6410C und 1/2-Zoll-Magnetbandeinheit.

Eine Weiterentwicklung des bekannten Minirechners FELIX I-102F (Rumänien) ist das Modell **I-102F/4M** mit einem 4-MByte-Hauptspeicher als bedeutendste Neuerung, aufgebaut aus 1-MByte-Platinen mit 64-KBit-ICs (Bild 11).
Als weitere Besonderheiten werden der Floppy-Disk-Controller (kompatibel zum IBM-3740-Format einfacher Dichte und zum PDP-Format doppelter Dichte) für 500 KBit/s sowie der auf einer Karte untergebrachte Magnetbandcontroller zur Zusammenarbeit mit der DMA genannt. Der Cache-Speicher hat eine Kapazität von 2 KByte und eine Zugriffszeit von 150 ns.
Als Betriebssysteme werden MIX und MININET – anwenderkompatibel mit RSX-11M und DECNET – angeboten.

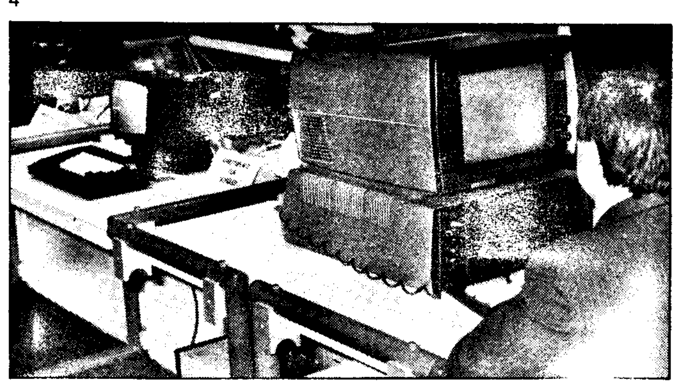
Auch die Republik Kuba stellte einige Erzeugnisse der Computertechnik vor – sowohl Eigenentwicklungen als auch in Lizenz gefertigte Produkte. Zum Lieferprogramm gehören außer Personalcomputern u. a. Tastaturen, Komponenten für lokale Netze, Terminals und ein umfangreiches Angebot an Software.
Die Computer werden entwickelt und produziert im Instituto Nacional de Sistemes Automatizados y Computacion (INSAC). Bei der Entwicklung spielt das ICID, eine Einrichtung des INSAC, eine große Rolle. Daher tragen die in Kuba entwickelten Computer in den Bezeichnungen das Kürzel CID. Der **CID 1408** (Bild 12) mit 8-Bit-Prozessor (8080) hat eine Speicherkapazität von 64 KByte RAM. Die Floppy-Laufwerke für das Gerät werden aus der VR Bulgarien geliefert. Als Betriebssystem wird CP/M verwendet. Gegenwärtig wird an der Entwicklung eines 16-Bit-PC (auf Basis des μP 8086) mit der Bezeichnung CID 1417 gearbeitet.

Es soll nur kurz erwähnt werden, daß zur Exposition Indiens auf der LFM der Computer **Eagle PC Plus** (Bild 13) gehörte, der voll kompatibel zum IBM PC XT ist – also Prozessor I 8088 (4,77 MHz), 256 KByte RAM, 2 Diskettenlaufwerke, Betriebssystem MS-DOS und wahlweise CP/M-86.
Das jugoslawische Unternehmen Iskra Delta offerierte zwei bemerkenswerte, im Leistungsspektrum weit auseinanderliegende Personalcomputer.
Das kleinere Modell, **Partner** (Bild 14), wurde in der Version WF/G mit einem Winchesterlaufwerk und einer Floppy-Einheit im Grundgerät gezeigt. Es handelt sich um einen grafikfähigen 8-Bit-PC (Z80A), der auch in einem lokalen Netz mit bis zu 8 PC als intelligentes Terminal eingesetzt werden kann. Als Betriebssystem wird CP/M verwendet.
Weitere Daten:
– 128 KByte RAM
– 5,25-Zoll-Hard-Disk mit 10 MByte, 5 MBit/s
– Minifloppy mit 682 KByte formatiert, 250 KBit/s
– Grafikbildschirm mit 26 Zeilen × 80 Zeichen bzw. 1024 × 512 oder 1024 × 256 Pixel
– Interface V.24, als Option Centronics.
Weit leistungsstärker ist das Modell **Trident** (Bild 15). Der Name beruht auf der Eigenschaft des PC, durch Austausch der Prozessorkarten mit drei verschiedenen Mikroprozessoren und damit auch unterschiedlichen Betriebssystemen arbeiten zu können. Die Architektur wurde an 8-, 16- und 32-Bit-Prozessoren angepaßt. Wählbar sind die Prozessoren 68010, APX 80286 und DEC J-11. Die CPU-Platinen enthalten außerdem z. B. eine MMU bis 16 KByte Adressierung, RAM von 256 KByte bis 1 MByte und Gleitkommaprozessor. Über den VME-Bus können weitere Moduln angeschlossen werden, so daß sich mehr als 1000 Kombinationsmöglichkeiten ergeben. Trident wird daher als universelles System bezeich-

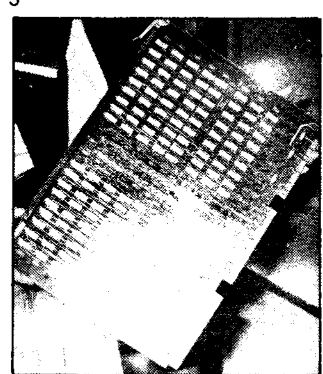
3



4



5



net, das sich sowohl als kommerzielles Multiusersystem als auch zur Prozeßautomatisierung oder als Grafikarbeitsplatz einsetzen läßt.
Der 14-Zoll-Farbbildschirm kann 640 × 480 Pixel sowie gleichzeitig 16 Farben von 256 möglichen darstellen.
Über 2-MByte-Schritte läßt sich der normale 512-KByte-RAM auf 16 MByte erweitern. 5,25-Zoll-Hard-Disks stehen von 40 MByte bis 80 MByte zur Verfügung, Floppy-Einheiten mit bis zu 1 MByte und Microstreamer bis zu 20 MByte.

Erstmals zu einer Messe in einem sozialistischen Land präsentierte das brasilianische Unternehmen CCE seine Produkte – unter anderem auch Computertechnik.
Das kleinste Modell ist der **EXECUTIVE XT** (Bild 16, links), ein portabler PC mit 8088-CPU (4,77 bzw. 8 MHz) und 8087-Koprozessor. In der Basiskonfiguration sind 8 KByte ROM und 704 KByte RAM, ein Minifloppylaufwerk (slimline), RS232C- und Centronics-Schnittstelle enthalten. Das Betriebssystem ist MS-DOS 2.10-kompatibel.
Ebenfalls IBM-kompatibel ist der **MC-5000XT** (Bild 16, Mitte) mit 8088- und 8087-Prozessor. Der RAM ist von 256 KByte auf 640 KByte erweiterbar. Die Grundvariante enthält ein slimline-Minifloppylaufwerk. Der Bildschirm hat 640 × 200 Pixel (einfarbig) bzw. 320 × 200 Pixel (mehrfarbig).
Kompatibilität zu Apple wird mit dem **MC-4000 EXATO PRO** (Bild 16, rechts) geboten (1-MHz-CPU 6502). Hervorzuheben ist die Vielzahl von Erweiterungskarten, unter anderem zur Nutzung des Z80 oder zur Vergrößerung des 48-KByte-RAM um jeweils 128 KByte. Als Betriebssysteme lassen sich CDOS, Apple DOS3.3, SUPERDOS, PRODOS und CP/M nutzen.

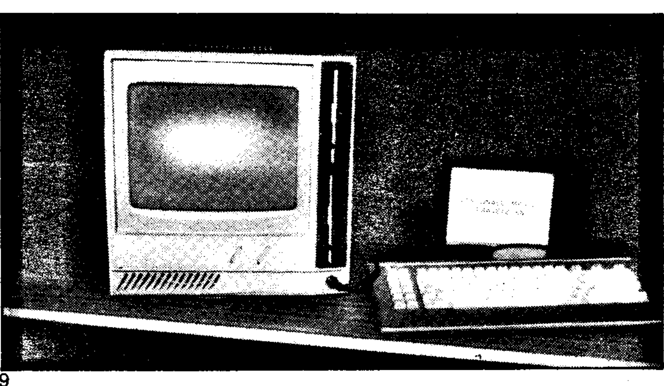
IBM zeigte in Leipzig den **PC AT3** (Bild 17), der unter den Betriebssystemen DOS 3.1. und Xenix läuft und drei- bis viermal schneller als der PC XT sein soll. Als Prozessor wird der I 80286 (16 Daten –, 24 Adreßbits, 8 MHz) eingesetzt. Weitere Leistungsparameter sind: 512 KByte RAM, 64 KByte ROM, Festplatte mit 30 MByte und Diskettenlaufwerk mit 1,2 MByte.

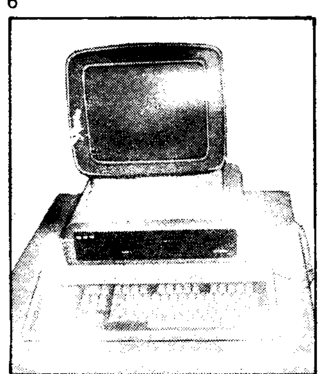
Die Handelsfirma TRANSCOMMERZ aus Berlin (West) stellt zu den Leipziger Messen Kopier- und Computertechnik aus. In erster Linie handelt es sich dabei um Erzeugnisse des Toshiba-Konzerns. Der kleinste PC-AT-Kompatible, so der Aussteller, sei der **Toshiba 3100** (Bild 18). Er hat in der Grundausstattung 640 KByte Hauptspeicher, der erweiterbar auf 4,6 MByte ist. Eine 3,5″-Festplatte mit 10 MByte und ein 3,5″-Diskettenlaufwerk mit 720 KByte Speicherkapazität sind in das Gerät integriert. Eine Reihe von Schnittstellen ermöglicht den Anschluß zusätzlicher Peripherie, z. B. externes Floppy-Laufwerk, Akustikkoppler, Farbdisplay.

EPSON ist seit einigen Jahren auf der LFM vertreten. Das Unternehmen hat eine sehr breite Produktionspalette und fertigt nahezu alle Komponenten für seine Finalerzeugnisse selbst. Zum Ausstellungsprogramm in Leipzig gehörten PC, sogenannte Handheld-Computer, Drucker und Baugruppen. Der **EPSON PC** (Bild 19) (Prozessor 8088, Koprozessor 8087 steckbar) läuft unter MS-DOS 3.2. 256 KByte RAM sind auf der Hauptplatine; erweiterbar auf 512 KByte, mittels einer Steckkarte auf 640 KByte aufrüstbar.
Den PC gibt es als Floppy-Disk-Version mit zwei Diskettenlaufwerken mit je 360 KByte und als Hard-Disk-Version mit einem Diskettenlaufwerk und 20 MByte Disk.
Der **HX-20** (Bild 20) gehört zur Gruppe der Handheld-Computer. Er verfügt über zwei 6301-CMOS-LSI-Mikroprozessoren und ein 32-KByte-ROM (intern auf 40 KByte, extern auf 72 KByte erweiterbar) und 16 bzw. mit Erweiterung 32 KByte RAM. Die als Option ins Gehäuse des HX-20 einsteckbare Mikrokassette speichert pro Seite (MC 30) 129 KByte. Am Gehäuse stehen eine RS 232 C (V.24) – Schnittstelle sowie ein High-Speed-Serial-Interface, ein Lesestift-Interface, ein Kassetten-Interface und der Systembus zur Verfügung. HX-20 kann auch Peripheriegerät wie Drucker, Kassettenspeicher oder Floppy Disk bedienen.
Das LCD-Display zeigt vier Zeilen mit je 20 Zeichen oder bei Grafik 120 × 32 einzeln ansteuerbare Punkte.

Commodore und ATARI waren in diesem Jahr erstmals auf der Leipziger Frühjahrsmesse vertreten. Beide sind als traditionelle Heimcomputerproduzenten bekannt. Immer mehr werden diese Geräte aufgrund ihrer Leistungsfähigkeit für betriebliche Aufgaben genutzt. ATARI stellte u. a. in Leipzig den **130XE** vor. Der 8-Bit-Computer hat eine Speicherkapazität von 128 KByte RAM und 24 KByte ROM. Die 128 K sind in zwei Blöcke zu je 64 K aufgeteilt. Über ein Bank-Switching ist der Zugriff möglich. Verwendet wird der Prozessor 6502 mit 1,79 MHz Taktfrequenz.
Eine Neuentwicklung ist der **ATARI PC** mit MS-DOS-Betriebssystem (Bild 22). Die Taktgeschwindigkeit ist von 4,77 auf 8 MHz umschaltbar. Der PC verfügt über 512 KByte Hauptspeicherkapazität und 256 KByte Bildschirmspeicher. Alle Peripheriegeräte der ST-Serie können angeschlossen werden.
Laut Commodore Büromaschinen GmbH soll der C 64 der meist verkaufte Computer der Welt sein. Eine Weiterentwicklung ist der **C 128 PC** (Bild 21). Er hat drei integrierte Prozessoren und somit drei Betriebssysteme. Im C-64-Modus können die Programme des C 64 abgearbeitet werden (6510-kompatibler Prozessor). Weiterhin gibt es den sog. C-128-Modus (Prozessor 8502, 1 oder 2 MHz) und den CP/M-Modus (Prozessor Z 80 A, 4 MHz). Die Speicherkapazität beträgt im C-64-Modus 64 KByte sonst 128 K.
I. Paszkowsky, H. Weiß
wird fortgesetzt in MP 6/87

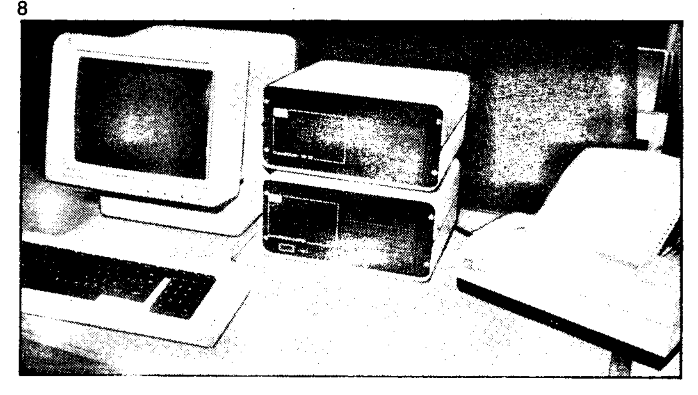
Fotos: Paszkowsky (11), Weiß (8), Werkfotos (3)


9

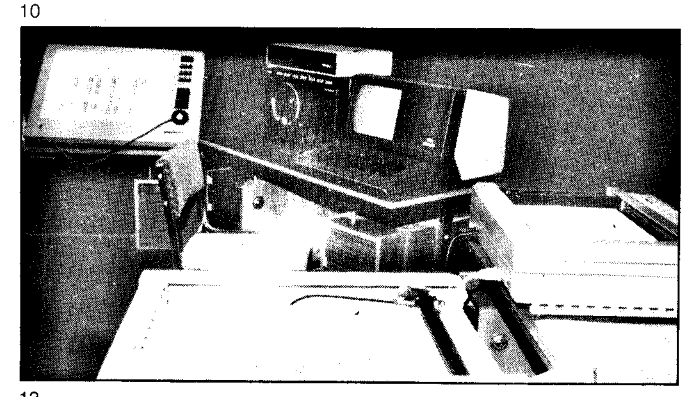

6



7


8

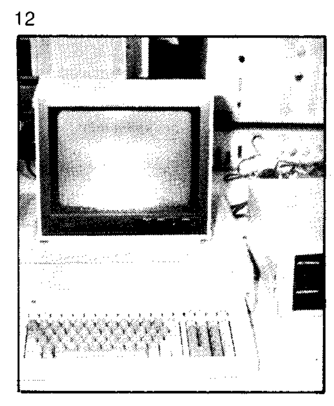
10



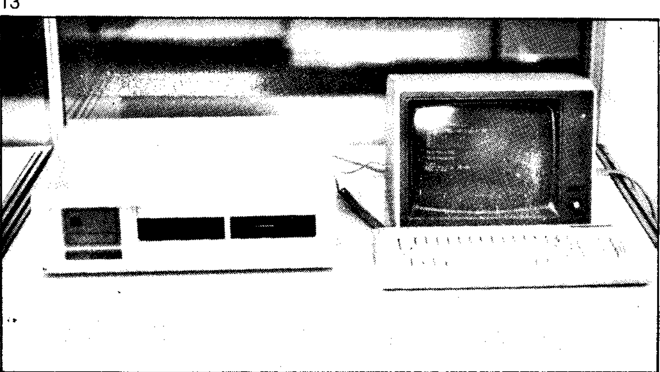
11



12

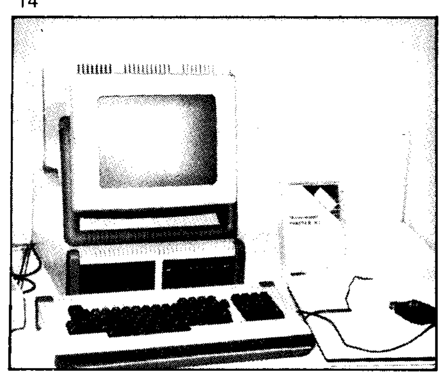


13

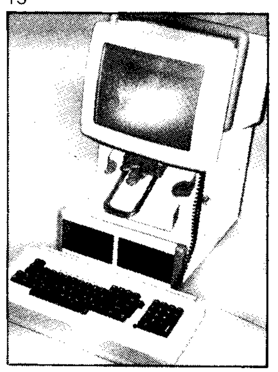




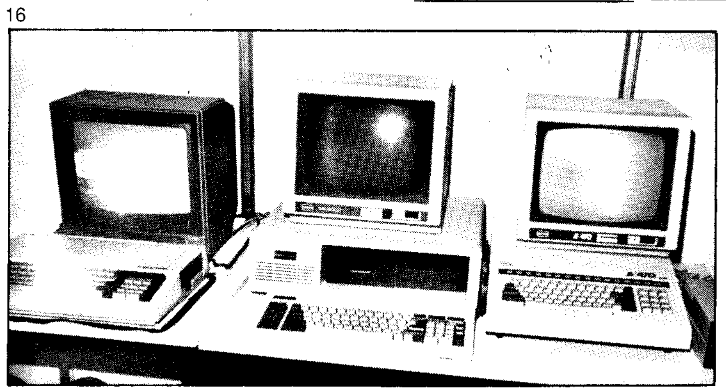
14



15



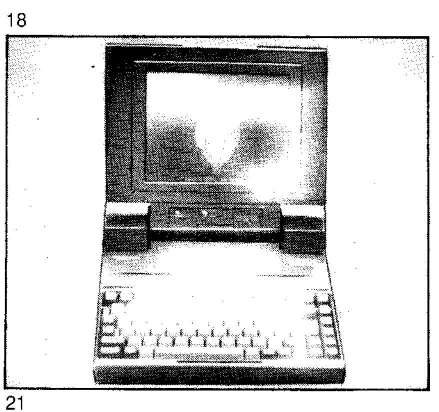
16



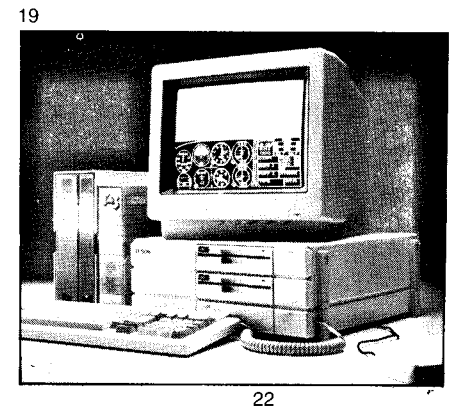
17



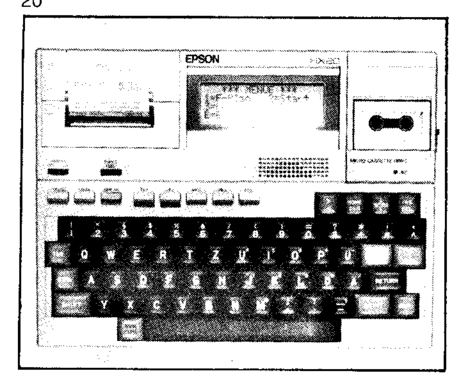
18



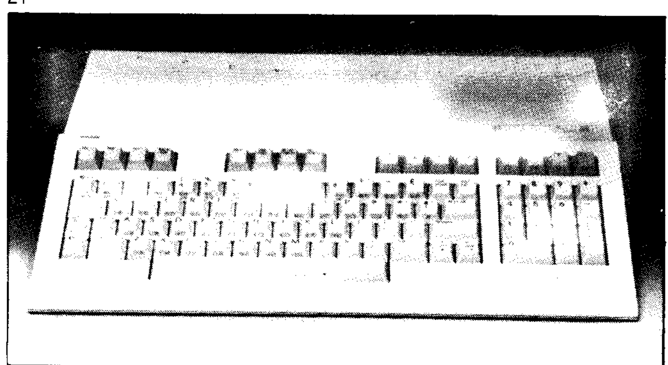
19



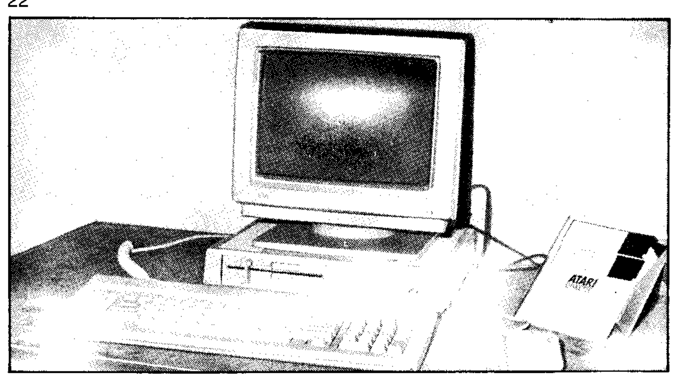
20



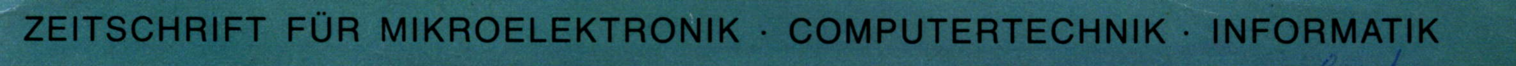
21



22



ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 5
S. 129–160



MP

Heft 6 · 1987

Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0233-2892

FORTH

Eine außergewöhnliche Softwarekonzeption

Hardware-Realisierung

Gleitkomma in FORTH

Gerätetreiber zum Nachladen

Terminalanschluß für Datennetze



Wissenschaft und Produktion der DDR im Dienste des Volkes

Die Ausstellung „Wissenschaft und Produktion der DDR im Dienste des Volkes", die sich in die Feierlichkeiten anläßlich des 750jährigen Bestehens von Berlin einfügt, zeigt anschaulich, wie die Werktätigen unserer Republik mit hohen schöpferischen Leistungen um die Lösung der Probleme bei der Entwicklung und Anwendung moderner Schlüsseltechnologien als wesentlichen Beitrag zur erfolgreichen Verwirklichung der Beschlüsse des XI. Parteitages der SED kämpfen.
In der Ausstellung wird deutlich gemacht, daß maßgebliche Voraussetzungen für das kontinuierliche und dynamische Wirtschaftswachstum in unserem Lande durch die immer engere Verbindung von Wissenschaft, Produktion und Absatz in den Kombinaten und Betrieben, ihre Kooperation auf dem Gebiet der Forschung und Technologie mit den Einrichtungen der Akademie der Wissenschaften der DDR, den Universitäten, Hoch- und Fachschulen geschaffen wurden und diese Verbindung weiter ausgebaut wird.
Die Ausstellung vermittelt ein lebendiges Bild von der Entwicklung und Anwendung der Mikroelektronik in allen Bereichen der Volkswirtschaft. Im Vordergrund stehen insbesondere moderne Lösungen der rechnergestützten Produktionsvorbereitung und -durchführung – CAD/CAM-Lösungen.
U. a. zeigt die Ausstellung die Entwicklung und Anwendung der flexiblen Automatisierung und der Robotertechnik. Es werden moderne Lösungen der Robotertechnik bei Automatisierungsvorhaben, vor allem im Maschinenbau, dargestellt.
Während der Ausstellung wird ein umfangreiches Vortragsprogramm durchgeführt, in dem Rundtischgespräche mit führenden Vertretern von Wirtschaft, Industrie und Wissenschaft, Fachvorträge und populäre Darstellungen bedeutender volkswirtschaftlicher und wissenschaftlich-technischer Prozesse vorgesehen sind.
Die Ausstellung findet in Berlin in der Werner-Seelenbinder-Halle in den Monaten Juni und Juli statt. Sie ist von 10.00 bis 18.00 Uhr geöffnet.

3. Angebotsmesse Wissenschaftlicher Gerätebau

Reges Interesse fand die 3. Angebotsmesse Wissenschaftlicher Gerätebau der Akademie der Wissenschaften der DDR, des Ministeriums für Hoch- und Fachschulwesen und der Akademie der Landwirtschaftswissenschaften der DDR, die Ende März an der Karl-Marx-Universität Leipzig veranstaltet wurde (Bild 1).
116 Exponate von 50 Einrichtungen der DDR wurden zur Nachnutzung angeboten. *ADN*

Zeit und Zugang kontrolliert mit ZEUSS

In vielen Betrieben und Institutionen ist es notwendig, den Zugang zu bestimmten Bereichen oder Einrichtungen zu kontrollieren und die Anwesenheit – oder auch Abwesenheit – von Personen zu registrieren – Aufgaben, die heute mit Computertechnik weitaus effektiver gelöst werden können als früher.
Während der Leipziger Frühjahrsmesse 1987 zeigte der VEB Robotron-Vertrieb Erfurt das Zeit- und Zugangskontrollsystem (ZEUSS) robotron A 5240, dessen autonom arbeitender Kern durch Anschluß an einen Verarbeitungsrechner und umfangreiche Peripherie zu einer leistungsfähigen Konfiguration ausgebaut werden kann (siehe Bild 2). Die über ein einadriges Koaxialkabel verbundenen Komponenten des Systems bilden dabei ein lokales Netz auf Basis des seriell ringförmigen Interfaces IFSR.
Als Masterterminal dienen PC 1715, BC A 5120 oder andere SCP-fähige Mikrorechner. Neben einem eigenständigen Systemterminal lassen sich bis zu 14 universelle Zeit-/Zugangsterminals mit eigener Zentraleinheit und verschiedenartiger Peripherie an den Master anschließen. Dazu können gehören: Handleseeinheiten für Magnetkarten (Plastkarten mit Magnetstreifen, als Personalkarten zusätzlich mit Foto), Lichtschranken, optische oder akustische Anzeigen, Tastatur für Sicherheitscodierung, Segmentanzeigen.
Als Zusatzausstattung (gehört nicht zum Systemangebot) sind Sperren, Drehkreuze oder Sicherheitstüröffner/-schließer einsetzbar.
Neben der Basissoftware gibt es ein Anwendungssoftware-Modulsortiment, mit dessen Hilfe z. B. Anwesenheitslisten, Sperrverzeichnisse, Zeitkonten, Prozeßprotokolle und Stammdaten erstellt werden können. *MP*



Gigabit-Chip in Entwicklung?

In etwa fünf Jahren wollen japanische Elektronik-Konzerne sogenannte Gigabit-Speicherchips fertigen, meldete kürzlich die BRD-Zeitung „Die Welt". Wie aus Forschungsberichten der japanischen Kyushu-Universität hervorgeht, soll dazu eine neue Technologie eingesetzt werden. Es handelt sich um die „Broch-Line"-Speicher, die in der VLSI-Technologie gefertigt pro cm^2 rund eine Milliarde Bit aufnehmen können. Die einzelnen Speicherzellen erfordern nur noch einen Platzbedarf von 0,5–1 mal 1–0,5 Mikrometer. Sie bestehen aus Halbleiter-Legierungen auf der Basis von Gadolinium und Gallium, die in netzartigen Strukturen aufgetragen und mit Polyimid-Harzen fixiert werden. Die enorm hohe Aufzeichnungsdichte ist physikalisch bedingt. Sie hängt damit zusammen, daß sich winzig kleine magnetische Domänen unter dem Einfluß elektrischer Ströme „orientieren" lassen. Das ist eine physikalische Erkenntnis, die gängig und akzeptiert ist, die allerdings im makromolekularen Bereich magnetischer Domänen bislang nicht genutzt werden konnte. Aufgrund der makromolekularen Strukturelemente erlaubt dieses Verfahren eine bisher unerreichte Speicherdichte digitaler Information. *ADN*

Pi auf 133 554 Millionen Kommastellen berechnet

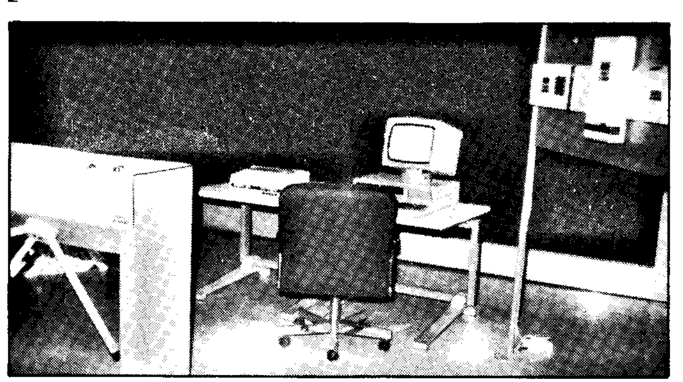
Der japanische Professor Prof. Yasumasa Kanada hat die Zahl „Pi" mit dem Computer bis auf 133 554 Millionen Stellen hinter dem Komma errechnet. Er arbeitete einen Monat am Programm für die Berechnung. Der Computer rechnete 37 Stunden lang und druckte das Ergebnis auf 20 000 Blättern Papier aus. *ADN*

Fotos: ADN/ZB (1), Weiß (1)

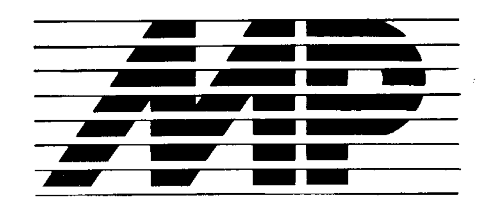
1



2



Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0233–2892



Mikroprozessortechnik, Heft 6 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 2 87 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)
Titel: Tatjana Stephanowitz

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

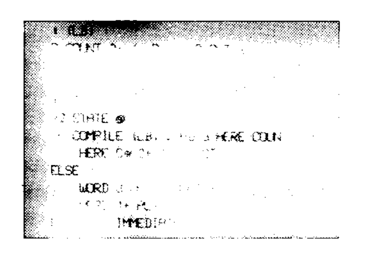
Redaktionsschluß: 23. April 1987

AN (EDV) 49837

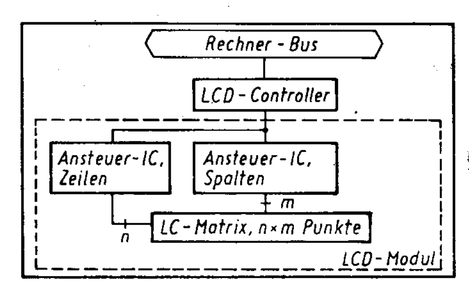
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

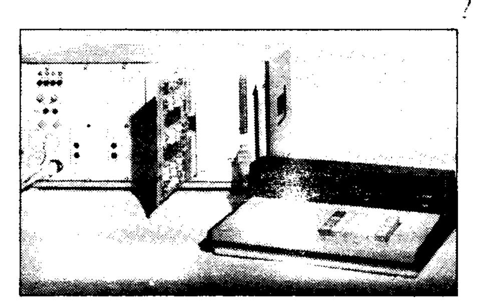
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustředni Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hà Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR–7010, und Leipzig Book Service, Talstraße 29, DDR–7010 Leipzig



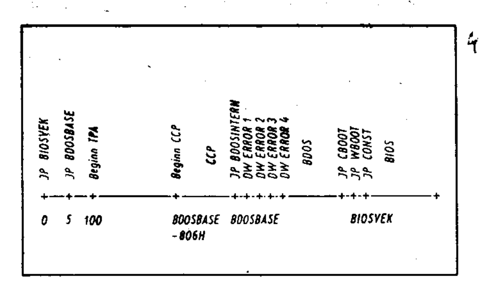
Seite 168



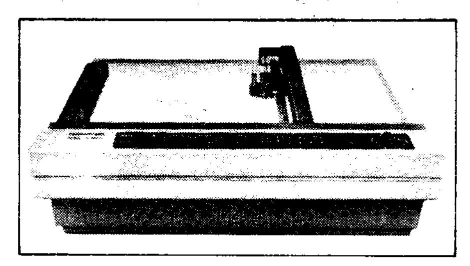
Seite 171



Seite 173 Foto: Tschacher



Seite 179



Seite 189 Werkfoto

Inhalt

Dialog

Lieber Leser,
auf die ersten Ausgaben der Mikroprozessortechnik haben die Redaktion viele Briefe erreicht, in denen Sie uns ihre Hinweise und Kritiken mitteilten.
Stellvertretend für die Vielzahl der Meinungen veröffentlichen wir nachfolgend einige Zuschriften, bevor wir in Heft 8/87 dazu ausführlich Stellung nehmen.

Ab Januar dieses Jahres habe ich ein Abonnement Ihrer Zeitschrift. Seit einigen Jahren interessiere ich mich für Mikroelektronik und Computertechnik. Auch beruflich kann ich das erworbene Wissen als Berufsschullehrer für das Fach Grundlagen der Automatisierung teilweise anwenden. Insbesondere für die Informatikausbildung der Lehrlinge war es notwendig, in sehr kurzer Zeit Grundlagen der Programmierung zu beherrschen. Seit einigen Jahren bin ich auch Leser der Zeitschrift Funkamateur und Radio, Fernsehen, Elektronik.
Mit großer Erwartung habe ich im vergangenen Jahr die Zeitschrift MP bestellt. Nach gewissenhaftem Studium der beiden ersten Hefte möchte ich Ihnen meine Meinung mitteilen.

Als positiv sind zu vermerken:

- grafische Gestaltung der Umschlagseite
- drucktechnische Aufbereitung
- Informationen und Dialogseiten
- Kontaktadressen zu den Beiträgen
- Kurs Programmierung in C
- Messeneuheiten.

Leider ist es z. Z. so, daß das Angebot an Fachbüchern über Programmiersprachen im Volksbuchhandel und in den Bibliotheken sehr bescheiden ist und in diesem Zusammenhang der Beitrag Kurs unbedingt fortgesetzt werden sollte.
Ich hoffe natürlich, daß eines Tages durch genügend Literatur diese Seiten anders genützt werden können.
Für die Planung der nächsten Hefte habe ich folgende Wünsche:

- Die hervorragende farbliche Gestaltung der Deckblätter sollte sich wenigstens teilweise im Innern der Zeitschrift fortsetzen.
- Die Messebeiträge und Info-Seiten sollten auch über Entwicklungen informieren, die den derzeitigen Welthöchststand repräsentieren.

Abschließend möchte ich feststellen, daß es wirklich sehr positiv war, eine Fachzeitschrift auf diesem Gebiet herauszugeben, da hierdurch konkrete Informationen zur Umsetzung der wirtschaftsentscheidenden Technologien gegeben bzw. durch Nutzung der Kontaktadressen beschaffbar werden. Ich bin zuversichtlich, daß sich in absehbarer Zeit Ihre Zeitschrift im Vergleich mit ähnlichen Zeitschriften der DDR eine Spitzenposition erkämpfen wird.

Manfred Heske, Werdau



Mit großem Interesse las ich die ersten Ausgaben dieser neuen Zeitschrift. Aufgrund des breiten Angebotes an Informationen habe ich mich entschlossen, mich mit einem Problem an Sie zu wenden.
Als Nutzer des PC 1715 und anderer Rechentechnik möchte ich diese natürlich für die vielfältigen Aufgaben, die sich aus meinem Studium und aus der Forschungsmitarbeit ergeben, einsetzen. Da ich besonders an der Assemblerprogrammierung interessiert bin, nutze ich die Übersetzerprogramme auf den verschiedenen Rechnern, z. B. SYS80 auf dem PC 1715. Doch da entstand ein Problem: An der Hochschule ist keine Möglichkeit der Einsichtnahme in Systemunterlagen des Betriebssystems SCP für den PC 1715 (speziell Systemunterprogramme – Ein/Ausgabe eines Zeichens auf dem Bildschirm –, Adressen der Ports, Adressen von Bildwiederholspeicher, CTC, DMA usw.) möglich, da entsprechende Unterlagen nicht existieren. Somit ist mir eine Assemblerprogrammierung kaum möglich. Da ja die Produktion von Personalcomputern die Stückzahl von 20000 bereits überschritten hat, sollte dies doch ein allgemein interessierendes Problem sein und deshalb eventuell ein Artikel darüber in Ihrer Zeitschrift erscheinen. Wenn dies nicht möglich ist, so wäre ich Ihnen dankbar, wenn Sie mir weiterhelfen können, wie ich entsprechende Informationen erhalten oder mir diese Informationen beschaffen könnte.
Ich möchte die Gelegenheit gleich nutzen, um Ihnen Lob und Tadel über die neue Zeitschrift zu übermitteln. Gut gefallen haben mir die Beiträge über den KC 85/3, die C-Programmierung und die Grafikmöglichkeiten mit dem PC 1715. Bei den Literaturhinweisen vermisse ich die Seitenanzahl und den Preis. Außerdem sollten nicht die 2. und 3. Umschlagseiten (Heft 2) Anzeigen vorbehalten sein! Besser ist es, darauf andere Artikel (siehe Heft 1) anzuordnen.

Thomas Kahl, Dresden

Ich bin im Ratiomittelbau im Halbleiterwerk Frankfurt(Oder) tätig und habe beruflich mit Mikroprozessortechnik zu tun (Einchipmikrorechner-Steuerungen). Mein Interesse an Computertechnik geht weit über die beruflichen Erfordernisse hinaus. Daher habe ich die MP abonniert. Die ersten beiden Hefte sind erschienen, und ich möchte kurz meine Meinung dazu darlegen. Auf den ersten Blick besticht die 1. Umschlagseite durch das Motiv und die farbliche Gestaltung. Bei näherem Hinsehen ist aber der Informationsgehalt äußerst gering, da das Bild nichts aussagt und nur die Überblicksinhaltsangabe Aufmerksamkeit verlangt. Die 2. und 3. Umschlagseiten von Heft 1 waren sehr gut. Konzentrierter Text mit technischen Daten, Fotos und Grafiken ergeben Anschaulichkeit und hohen Informationsgehalt. Dagegen finde ich die im Heft 2 nicht geeignet. Die 4. Umschlagseiten sind gut, wobei der Informationsgehalt der von Heft 2 deutlich geringer ist als in Heft 1. Ausgezeichnet gefällt mir der Aufbau der einzelnen Beiträge. Sie sind sehr übersichtlich angelegt und hoch informativ. Besonders lobenswert ist, daß die Autoren kurz in Wort und Bild vorgestellt werden und die jeweilige Kontaktadresse angegeben ist. Die Rubriken „Info", „Kurs", „Bericht", „Literatur" sowie „Börse" sind nach meiner Ansicht gut angelegt und sollten in dieser Form beibehalten werden.
Nicht ganz einverstanden bin ich mit dem Inhalt einiger Beiträge. Zum Beispiel ist der Artikel „CMOS-Gate-Array-System U5200" aus Heft 1 fehl am Platze und sollte eher in der rfe erscheinen. Dafür sollte Softwareproblemen mehr Platz eingeräumt werden.

Bernd Thiel, Frankfurt(Oder)

Herzlichen Glückwunsch zu den ersten Ausgaben der MP. Das Äußere ist sehr ansprechend, weiter so!
Zum fachlichen Inhalt würden wir begrüßen, wenn noch umfangreichere Informationen und Artikel, die nutzerorientiert sind, z. B. Programmbeschreibungen und Listings der unterschiedlichen Computertypen, in der Folgezeit veröffentlicht würden. Die Serie Kurs in der MP (Programmierung in C) gefällt uns gut, und wir hoffen, daß der Platz auf den Innenseiten beibehalten bleibt, es vereinfacht die zusammenhängende Archivierung solcher Folgen.

Computer-Club
Robotron-Anlagenbau Leipzig

Das im Artikel *„Arbeit mit BASIC-Datenfeldern beim KC 85/3"* des Märzheftes der MP angesprochene Problem der Verwendung von variablen Dateinamen bei Kassettenoperationen läßt sich relativ einfach lösen.
Der Interpreter erwartet nach der Ladeanweisung den in Anführungszeichen gesetzten Dateinamen. Bieten wir ihm nun hier das als erstes erwartete Anführungszeichen in Form eines leeren Strings und verknüpfen diesen String mittels Pluszeichen mit einer Stringvariablen, die den Dateinamen enthält, so erledigt die Stringauswertung die restliche Arbeit für uns. Kassettenoperationen lassen sich somit problemlos variabel an allen Stellen eines Programms einbinden und ebenso im Direktmodus verwenden.

CSAVE*""+N$;A$	Laden und Speichern von
CLOAD*""+N$;A$	Datenfeldern mit dem Namen, der in N$ enthalten ist
CSAVE""+N$	Laden und Speichern von
CLOAD""+N$	Programmen mit dem Namen, der in N$ enthalten ist.

Inwieweit eine Verwendung eines variablen Dateinamens bei Kassettenoperationen im Direktmodus sinnvoll ist, sei dahingestellt. Mit diesem simplen Trick vermeidet man die im angesprochenen Artikel verwendete Methode des „Hineinpokens" in das BASIC-Programm, eine Methode, die gerade für Anfänger mit Vorsicht zu genießen ist. Getestet wurde das vorgeschlagene Verfahren mit dem KC 85/2 und Modul 006. Jedoch dürfte es auch bei anderen Versionen des KC funktionieren.

M. Müller, Berlin

Prüfe ob auch INPUT ""+N$: N$

FORTH: Eine außergewöhnliche Softwarekonzeption

Gert-Ulrich Vack
Zentralinstitut für Kybernetik
und Informationsprozesse
der AdW der DDR

Programmiersprachen für Mikrorechner

In den ersten Jahren des Einsatzes von Mikrorechnern erfolgte deren Programmierung in Maschinencode oder Assemblersprache. Auf Grund des wachsenden Leistungsvermögens der Prozessoren und der dadurch zunehmenden Komplexität der Programme war der Übergang zur Programmierung in höheren Programmiersprachen zwingend notwendig. Dabei griff man überwiegend auf Sprachfamilien zurück, die sich in den Jahren zuvor auf Kleinrechnern oder EDVA mittlerer Größe durchgesetzt hatten. Diese Programmiersprachen sind geprägt von Randbedingungen hinsichtlich Einsatzgebiet (z. B. numerische Verarbeitung großer Datenmengen mit FORTRAN), durch Rechnerhersteller (PL/1 für IBM-Systeme, C für PDP-Kleinrechner), Abarbeitungsregime (überwiegend im Stapelbetrieb) und Trägerbetriebssysteme (z. B. C unter UNIX). So dominieren gegenwärtig bei Mikrorechnern die Programmiersprachen PL/M bzw. PL/Z, PASCAL, C, FORTRAN sowie für Spezialzwecke LISP und PROLOG.

Anforderungen an Programmiersprachen und -systeme für Steuerungsaufgaben

Der überwiegende Teil der Mikrorechner wird für Steuerzwecke eingesetzt: in Fahrkartenautomaten, Robotern, bei der Experimentautomatisierung bis hin zur Großautomatisierung industrieller Prozesse (flexible Fertigungssysteme), um nur einige Beispiele zu nennen.
Für die Programmierung gelten (z. B. im Unterschied zu Kleinrechnern) z. T. wesentlich andere *Randbedingungen,* die vor traditionellen Programmiersprachen in Einheit mit den zur Verfügung stehenden Programmierumgebungen (das sind alle für die Programmierung benötigten Hilfsmittel wie Editor, Compiler, Debugger) nicht oder nur teilweise erfüllt werden:

- Gestaltung der Software unter Aspekten eines Echtzeitsystems
- vollständige oder weitestgehende Beherrschung und Programmierbarkeit der Hardware-Ressourcen (einschließlich z. B. Interruptsteuerung, Einzelbytezugriffe im Speicher, Direktadressierung von Ein-/Ausgabetoren)
- speicherplatz- und laufzeitoptimale Codegenerierung für die Anwenderprogramme (auch unter den heutigen Gegebenheiten billiger Halbleiterspeicher ist bei einer Massenproduktion, wie sie z. B. die Automatisierungstechnik erfordert, aus Zuverlässigkeits- und Preisgründen nach wie vor ein minimaler Speicherbedarf anzustreben)
- nutzerfreundliche, hochgradig interaktive Programmierumgebungen, die vor allem den Integrationstest (Zusammenbinden von Hardware- und Softwarekomponenten etwa in einem Meßgerät) unterstützen
- leichte Modifizierbarkeit und Erweiterungsfähigkeit, hoher Grad an Modularisierung (um auf Fehlersituationen, Betriebserfahrungen und neue Einsatzrandbedingungen flexibel reagieren zu können).

Auch in der DDR werden in großen Stückzahlen Mikrorechner verschiedener Typen in der Volkswirtschaft eingesetzt. Damit stellt sich die Frage nach geeigneten Programmiersystemen, die den genannten Forderungen gerecht werden. Zusätzlich ist eine Portabilität der Software (d. h. deren unkomplizierte Übernehmbarkeit auf andere Rechnertypen oder Betriebssysteme) ein ökonomisches Erfordernis, um die Wiederverwendbarkeit von Programmen zu sichern. (Die Software ist zum kostenbestimmenden Faktor bei der Erarbeitung von Mikrorechner-Systemlösungen geworden).

Anwendererfahrungen belegen, daß mit traditionellen Programmiersprachen und Softwarekonzepten diese Forderungen z. T. nur bedingt zu erfüllen sind. Hinzu kommt, daß der Anwender eines Mikrorechners in den Begriffen seines Fachgebietes denken möchte. Jede „gedankliche Umschaltung" zwischen dieser Begriffswelt und den Termini, Einsatzbedingungen und Einschränkungen traditioneller Softwareentwurfssysteme bedeutet einen Verlust an Kreativität – direkt ausdrückbar in geringerem Leistungsvermögen der Software und damit des Rechners bzw. höherem Entwurfsaufwand. Letztlich entsteht daraus ein ökonomisches Problem: Wie schnell und mit welchem Aufwand kann möglichst effektiv eine Softwarelösung erarbeitet werden? Die Antwort darauf bestünde in der Gestaltung von Fachsprachsystemen mit entsprechenden Eigenschaften. Dafür stehen jedoch nur wenige oder keine Entwurfsmittel zur Verfügung. So sind viele Systementwickler gezwungen, ihre Problemstellung in den vorgegebenen Rahmen einer universellen Programmiersprache zu pressen, die ursprünglich für andere Zwecke konzipiert war. Erweiterungen dieser Programmiersysteme um Funktionen für eine spezielle Aufgabenklasse durchbrechen häufig deren konzeptionellen Rahmen, führen zu Uneinheitlichkeiten in der Nutzung der Software (z. B. verschiedene Kommandosprachen, die unterschiedliche Syntax aufweisen) oder zu „Behelfslösungen" und erschweren zudem die Portabilität.

Entstehung und Verbreitung von FORTH

International und national findet zunehmend ein Programmier- und Systemkonzept Verbreitung, welches die eingangs genannten Probleme zu lösen vermag, dabei aber einen weitestgehenden Bruch mit traditionellen Vorstellungen und Erfahrungen in der Programmierung vollzieht bzw. konzeptionell z. T. bereits das vorwegnimmt und realisiert, was für herkömmliche Programmierumgebungen zunächst noch Zielstellung ist und in Form von Zusätzen mühevoll erarbeitet werden muß. Dabei gehört FORTH unter den mehr als 1000 Rechnersprachen zu den wenigen Dutzend Lösungen, die sich in der Praxis behaupten können und permanent weiterentwickeln. FORTH ist eine lebende Sprache.
In den 70er Jahren konzipierte Charles Moore (USA) FORTH, und zwar nicht, um eine weitere Programmiersprache zu kreieren, sondern um sich ein effektives Werkzeug zur Lösung seiner Softwareaufgaben zu schaffen. Herkömmliche Sprachkonzepte erwiesen sich für eine effektive anwendungsspezifische Programmierung von Echtzeitsystemen als zu starr.
Nach einer Anlaufphase, in der Anwendungen vor allem in der Radioastronomie erfolgten, verbreitete sich FORTH zusehends und wird heute in Zusammenhang mit Aufgabenstellungen wie Raumfahrt, Robotertechnik, Automatisierungstechnik, Bildverarbeitung, Meßtechnik, Sensortechnik, Künstliche Intelligenz, Medizintechnik oder Datenbanksystemen genannt. Auch im Bereich der Heimcomputer hat FORTH viele Anhänger gefunden, die damit Spielprogramme, Grafik- oder Musiksyntheseprogramme erstellen[1].

Implementierungsgrundlagen und Nutzungsbesonderheiten

Folgende wesentliche *Implementierungskonzepte* werden in FORTH umgesetzt:

- interne Codierung von Programmen im Fadencode (das sind Sequenzen von Adreßzeigern zu Codefragmenten, ähnlich wie Folgen von Unterprogrammaufrufen ohne Codierung des CALL-Befehls)
- Stackmechanismen für implizite Parameterübergaben (Parameterstack) und die Steuerung der Aufrufhierarchie von Moduln (Returnstack)
- Wörterbuchkonzept zur Verwaltung der Schlüsselwörter und der zugehörigen Funktionen, einheitliche Darstellungsform für Objekte und Operationen

---

[1] An dieser Stelle sei auf einen Beitrag verwiesen, den wir zum Thema „FORTH auf Kleincomputern" für Heft 8/87 vorgesehen haben. Red.

- virtuelle Speichertechnik zur Verwaltung des Massenspeichers.

Daraus sowie aus weiteren Eigenschaften von FORTH resultieren folgende *Nutzungsbesonderheiten:*

- FORTH-Programme sind hochgradig modularisiert (bottom-up-Implementierung des top-down-Lösungskonzepts) und streng strukturiert (GOTO-Konstruktionen sind nur in Ausnahmefällen erlaubt).
- Verzicht auf Klammerschreibweisen, Darstellung von Operanden und Operationen in umgekehrt polnischer Notation (d. h., Operatoren stehen nach den Operanden)
- Parameterübergaben zwischen Moduln erfolgen in der Regel implizit über den Parameterstack als Kommunikationsmedium (es sind aber auch Variablen zulässig).

Nutzungseigenschaften

Mit diesen Prinzipien gelang es, ein Programmiersystem mit folgenden Eigenschaften zu realisieren:

- FORTH ist ein interaktives, integriertes Programmiersystem.

Vergleichbar mit TURBO-PASCAL bietet FORTH die Möglichkeit, alle zur Programmeingabe (Editor), Übersetzung (Assembler, Compiler), Testung (Debugger) und Rückübersetzung (Discompiler) notwendigen Komponenten gleichzeitig im Speicher zu laden; dafür genügen etwa 15 ... 25 KByte. Alle Komponenten der Programmierumgebung unterliegen einem einheitlichen sprachlich-syntaktischen Konzept. Gegenüber TURBO-PASCAL hat FORTH jedoch u. a. den entscheidenden Vorteil, interpratativ wie BASIC arbeiten zu können: das heißt, alle eingegebenen Zeichenketten werden sofort von einem Wortinterpreter analysiert und die entsprechenden Funktionen aktiviert.

- FORTH realisiert elementare inkrementelle Compilationsmechanismen und wird damit zum erweiterbaren Programmiersystem.

Um den Interpretationsoverhead auszuschalten, kann eine Funktion als Quelltext eingegeben und dabei in den internen Fadencode als Erweiterung des Wörterbuches übersetzt werden. Das ermöglicht die schrittweise Erweiterung des Bestandes an Schlüsselwörtern und Funktionen im Wörterbuch in Richtung der gewünschten Anwendungsfunktion – bei gleichzeitiger Sicherung der Konsistenz des Gesamtsystems und unter Beibehaltung der Eigenschaften an der Nutzerschnittstelle.

- FORTH ist ein interaktives System. Die Systemkomponenten sind echt im Dialog nutzbar, das Wirtssystem ebenso wie die damit und darin implementierte Anwendersoftware. Das gestattet es insbesondere, alle Funktionen des FORTH-Systems sowie die bereits implementierte Anwendersoftware für die weitere Algorithmenfindung und -implementierung zu verwenden.

FORTH als Fachsprachprogrammiersystem

Programmieren in FORTH ist die Erweiterung des Funktions- und Datenbestands in Richtung der Problemstellung mittels eines inkrementellen Compilationsmechanismus. Dabei löst sich der Programmierer schrittweise von den Elementarfunktionen des FORTH-Systems und kann sich mehr und mehr auf die für seinen Anwendungsfall typischen Funktionen beziehen (das sind „Befehle" wie VENTIL AUF SPEICHER LOESCHEN oder BILD ZEIGEN).

Um ein Anwendungsproblem lösen zu können, muß man den Vorrat an vordefinierten FORTH-Funktionen entsprechend und in Richtung der Aufgabenklasse erweitern: um neue Datentypen (Klasse von Operanden), neue Funktionsklassen und neue Funktionen.

Diese neuen Elemente werden Bestandteil des integrierten Programmiersystems und erweitern damit FORTH fachsprachlich, wobei die Implementierungs- und Sprachkonzeptionen erhalten bleiben. Der Nutzer greift auf die Anwendersoftware in gleicher Weise zu wie auf das FORTH-System.

Programmiermethodik in FORTH

In FORTH implementiert man z. T. wesentlich anders und dabei aber auch beträchtlich effektiver als in anderen Programmierumgebungen für traditionelle Programmiersprachen. Man entwirft zunächst meist ein Grobkonzept („60-%-Ansatz"). Dabei ist FORTH als Beschreibungssprache nutzbar – mit der Besonderheit, daß FORTH-Pseudocode mühelos in ein abarbeitbares Programmskelett überführbar ist (z. B. bestehend aus den wichtigsten Steuerstrukturen). Den eigentlichen Detailentwurf nimmt man gemeinsam mit der Implementierung nach der Methode der schrittweisen Verfeinerung vor. Dabei baut man Algorithmus und Programm allmählich zum vollen gewünschten Funktionsumfang aus.

Durch den hohen Modularisierungsgrad und die dadurch bedingte relativ geringe Komplexität der Teilfunktionen bleiben die Implementierungsprobleme überschaubar.

Die integrierte FORTH-Programmierumgebung mit allen bereits entwickelten Anwendersoftwarekomponenten steht für die Erarbeitung der Mini-Algorithmen einzelner Module und zur Implementierung als FORTH-Funktionen (ebenso wie Datenobjekte werden diese in FORTH als Wörter bezeichnet) zur Verfügung. Charakteristisch ist, daß nach einer genauen Problemanalyse beim Programmentwurf wesentlich intensiver mit dem Rechner gearbeitet wird als in anderen Systemen. Dabei werden Entwurfsideen in einer realen Softwareumgebung erarbeitet und erprobt.

Auf Grund der geringen turn-around-time (Zyklus zwischen Programmeingabe, -compilation, Entwurf von Testfunktionen, Testung; letztlich also die Zeit von der Idee bis zum laufenden Programm) sowie in Verbindung mit der durch die Erweiterbarkeit von FORTH bedingten Flexibilität und der Problembezogenheit im Sinne eines Fachsprachensystems erreicht man eine bis zu zehnmal höhere Programmierproduktivität als in anderen Systemen.

Eigenschaften von FORTH-Programmen

FORTH-Programme sind rasch implementierbar, werden schnell abgearbeitet (bei 16-Bit-Systemen etwa 20 ... 100 % langsamer als Assemblerprogramme), sind durch kurze Quelltexte darstellbar und haben nur geringen Speicherbedarf.

Typisch sind folgende Parameter: 1 FORTH-Wort (vergleichbar mit 1 Unterprogramm zur Realisierung einer Teilfunktion) umfaßt etwa 2,5 ... 5 Zeilen Quelltext und wird in etwa 25 Byte compiliert. Die Implementierungseffektivität (einschl. Entwurf, Codierung, Test) beträgt bei geübten FORTH-Programmierern etwa 65 Zeilen je Tag. Typische Programmlängen sind einige KByte bis einige 10 KByte. Ein an Wordstar orientierter, auf FORTH zugeschnittener Screeneditor (mit etwa 40 Kommandos) belegt 3,5 KByte Programm zuzüglich FORTH-System und ist in 450 Zeilen Quelltext einschließlich Menüs notiert.

Erweisen sich FORTH-Programme doch einmal als zu langsam, so ersetzt man die wenigen zeitkritischen Teile durch In-Line-Assemblerprogramme.

FORTH und Fremdprogramme

Fremdprogramme sind Softwaremoduln, die in anderen Programmiersprachen geschrieben sind und z. T. bereits als Lademoduln oder Objektbibliotheken zur Verfügung stehen. Das Zusammenbinden mit FORTH-Programmen ist problemlos. Erfahrungen liegen zur Einbindung von PASCAL-, FORTRAN- und Assemblerprogrammen vor. Damit bleibt der in diesen Sprachen existierende Software-Fundus weiterhin nutzbar und kann in die interaktive, integrierte FORTH-Programmierumgebung einbezogen werden.

FORTH in der DDR

Nach ersten Veröffentlichungen in der DDR im Jahr 1981 liegen inzwischen praktische Einsatzerfahrungen vor. Zentren der Entwicklung und Anwendung von FORTH-Systemen waren und sind u. a. das Zentralinstitut für Kybernetik und Informationsprozesse der AdW der DDR, die Wilhelm-Pieck-Universität Rostock (Sektion Technische Elektronik) sowie die Technische Hochschule Ilmenau (Sektion Technische und Biomedizinische Kybernetik).

Wurden bisher Systeme eingesetzt, die zum Standard FIG-FORTH der FORTH Interest Group (ähnlich dem Standard FORTH-78) kompatibel waren, so vollzieht sich mit dem Einsatz der 16-Bit-Technik gegenwärtig der Übergang zum 1983 vom FORTH-Standards-Team definierten FORTH-83.

Zur besseren Koordinierung der Softwarearbeiten und zur Sicherung des Informationsaustauschs der Anwender und Entwickler wurde im Bezirksverband Suhl der KDT die Interessengemeinschaft FORTH gegründet. Alle Interessenten und Anwender von FORTH können diese IG unter folgender Anschrift erreichen: Kammer der Technik, Interessengemeinschaft FORTH, Postschließfach 190, Ilmenau, 6300.

Die IG FORTH möchte über Grundlagenliteratur und deren Benutzungsmöglichkeiten informieren, in der DDR eingesetzte FORTH-Systeme vorstellen, Erstanwender in geeigneter Form bei der Einarbeitung in FORTH unterstützen und die Erfahrungen aller FORTH-Anwender weitergeben. Angestrebt wird, in Form von KDT-Arbeitsmaterialien u. a. Standards und verfügbare Systeme zu publizieren.

Zusammenfassung

In diesem Beitrag wurde ein Überblick über wesentliche Aspekte der Programmierung in FORTH gegeben; eine Einführung in die Programmiersprache muß späteren Veröffentlichungen vorbehalten bleiben.
FORTH läßt sich am besten charakterisieren durch Begriffe wie Methodik, Konzept und Technologie der Softwaregestaltung, Programmiersprache, Betriebssystem, integrierte interaktive Programmierumgebung. Im Unterschied zu anderen Systemen ist FORTH ein Toolkit aus einfachen, wenig komplexen, dafür aber sehr leistungsfähigen Programmierwerkzeugen und ermöglicht einen problembezogenen Softwareentwurfsprozeß. Die Integration von Softwareentwicklungssystem und Anwendungssoftware unterstützt eine prototyporientierte Programmimplementierung. Für umfangreiche numerische Operationen weniger prädestiniert, ist FORTH hervorragend auf die Belange von Steuerungsproblemen ausgerichtet, wobei der Begriff auch die Steuerung von Softwaresystemen einschließt.
In den kommenden Jahren ist mit einer weiteren Verbreitung von FORTH zu rechnen. Unter den spezifischen Randbedingungen des Einsatzes von Mikrorechnern in der DDR kann damit ein Beitrag zur Lösung softwaretechnologischer Aufgaben geleistet werden.

Literatur

/1/ Brodie, L.: Programmieren in FORTH. Coedition von Carl Hanser Verlag und Prentice-Hall International. München, Wien, London 1984
/2/ Rather, E. D.: Forth takes aim at real-time applications. COMPUTER DESIGN Oct. 15, 1986, S. 85–90
/3/ Glaser, F.: Flexible Programmiersprache für die Automation. ELEKTRONIKSCHAU (1986) 12, S. 56–59
/4/ Brodie, L.: Thinking FORTH. Prentice-Hall International London, 1984
/5/ A Bibliography of FORTH References. 2. Ausgabe. Oktober 1984; Institute for Applied FORTH Research, Inc.
/6/ FORTH-83 Standard. Mountain View Press, August 1983
/7/ Der Standard FORTH-83. (in Vorbereitung bei IG FORTH)
/8/ Vack, G.-U.: Mikrorechnerprogrammierung in der Dialogsprache „FORTH“. Vortrag auf dem 27. Internat. Wissensch. Kolloquium der TH Ilmenau 1982. Vortragsreihe „Entwurf, Analyse und Einsatz von informationsverarbeitenden Geräten und Systemen“.

Ein fig-kompatibles FORTH für den U8000

Benno Schiemann, Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Abteilung Rechentechnik und Wissenschaftlicher Gerätebau

Zur Nutzung einer 16-Bit-ZRE mit dem Prozessor U8000 /1, 2/ für Steuerungs- und Automatisierungsaufgaben wurde nach einer geeigneten Hochsprache gesucht. Ausgehend von den Erfahrungen, die an der IH für Seefahrt Warnemünde/Wustrow mit der Programmiersprache FORTH bei der Implementierung und Anwendung auf 8-Bit-Systemen gemacht wurden, fiel die Entscheidung für diese Sprache. Damit wurden auch die guten Ergebnisse, die an der Wilhelm-Pieck-Universität Rostock mit FORTH erreicht werden konnten, berücksichtigt /3/.

FORTH-Eigenschaften

FORTH ist eine Programmiersprache mit einer übersichtlichen und für Erweiterungen offenen Struktur. FORTH ist zu etwa 80 Prozent selbst in FORTH geschrieben (also lediglich 20 Prozent in Assembler). Der FORTH-Kern ist mit etwa 7 KByte Speicherbedarf relativ klein. Diese Fakten begünstigen die gegenüber anderen Hochsprachen vergleichsweise schnelle Implementierung von FORTH auf den verschiedensten Prozessoren.
Die interaktive Arbeitsweise in FORTH bringt Vorteile bei der Programmerstellung, -testung und -wartung.
FORTH bietet ein unkompliziertes Konzept der Massenspeicherverwaltung. Der Massenspeicher wird als eine Erweiterung des vorhandenen Speichers behandelt (virtueller Speicher). Der Zugriff erfolgt über ein „Fenster“, das im Adreßbereich des virtuellen Speichers „verschoben“ werden kann.
Der FORTH-Kern ist durch den Anwender für seine Zwecke nahezu beliebig erweiterbar, wodurch sich FORTH als ein universelles, leistungsstarkes Software-Werkzeug präsentiert, das dem Nutzer die Möglichkeit bietet, sich seine eigene, den spezifischen Bedürfnissen angepaßte Hochsprache zu schaffen. Das macht gleichzeitig eine der Stärken von FORTH aus.
FORTH ist mit all diesen Eigenschaften nicht nur *Hochsprache,* sondern kann auch als *Betriebssystem* (FORTH hat, wie bereits erwähnt, eine eigene Massenspeicherverwaltung.) und als ein umfassendes *Software-Konzept* angesehen werden.
FORTH unterstützt solch zeitgemäße Verfahren der Softwaretechnologie wie strukturierte, modulare Programmierung und Top-down-Planen/Bottom-up-Programmieren.
In FORTH geschriebene Programme sind kompakt und relativ schnell (je nach Programmierung 50 ... 200 Prozent time-overhead gegenüber gleichwertigen Assemblerprogrammen /4/). FORTH- und Assemblerprogramm-Module sind beliebig mischbar.
Die Maschinennähe von FORTH zeigt sich u. a. in der Möglichkeit, direkt auf Rechnerressourcen (Speicher, Ports) zurückgreifen zu können. Typisch für FORTH sind die umgekehrt polnische Notation (UPN), eine Listenstruktur (FORTH-Wörterbuch) und das Stapelprinzip (Datenstack, Parameterstack) /5/.
Das Stapelprinzip ist auch bei der Mehrzahl der FORTH-Worte (der Begriff Wort steht in FORTH für Programm-Modul) in Form eines Adreß-Stapels wiederzufinden – eine Eigenschaft, die bei der hier vorgestellten U8000-Implementierung vorteilhaft ausgenutzt wird.

U8000-FORTH-Version

Direkte Ausgangspunkte für das U8000-FORTH waren eine Programmliste des 8080-fig-FORTH und eine Veröffentlichung über ein Z8000-FORTH /6/, das jedoch starke Abweichungen vom fig-Standard aufweist und nur einen eingeschränkten Wort-Umfang besitzt. Die hier vorgestellte Version realisiert den vom fig-Standard her bekannten Wort-Umfang und belegt etwa 7 KByte. Das U8000-FORTH arbeitet nichtsegmentiert.
FORTH-Schnittstellen zur Umgebung sind:

– Tastatur einschließlich STATUS über die Worte **KEY** und **?TERMINAL**
– Bildschirm über das Wort **EMIT**
– Drucker mit ^P einschaltbar und dann über **EMIT** erreichbar
– Massenspeicher über das Wort **R/W**.

Diese Schnittstellen nutzen in der vorgestellten Version wiederum als Schnittstelle System-Calls des Prozessors U8000 (soweit vom übergeordneten System unterstützt) für den Zugriff auf die genannten Systemkomponenten.
Die Programmiersprache FORTH bezieht sich auf einen hypothetischen 16-Bit-Stackcomputer mit entsprechenden Registern und Stapelzeigern. Diese Register und Zeiger müssen auf der konkreten Maschine durch deren Register bzw. durch Speicherzellen nachgebildet werden. Der komfortable Registersatz des U8000 mit seinen Universalregistern (die ja u. a. auch als Stackpointer fungieren können) unterstützt eine einfache Nachbildung der entsprechenden FORTH-Register und -Zeiger.
Die vorliegende Implementierung nutzt hierfür folgende U8000-Register:

– R1 Instruction-Pointer IP der „FORTH-CPU“
– R2 Word-Address-Register WA der „FORTH-CPU“
– R3 Temporäres Register T der „FORTH-CPU“
– R14 Return-Stackpointer RSP
– R15 Daten-(bzw. Parameter-)Stackpointer.

Der Gebrauch dieser Register in FORTH wird anhand der Routinen NEXT und DOCOL in Tafel 1 dargestellt. Gleichzeitig ist in dieser Tafel eine Gegenüberstellung dreier FORTH-Versionen vorgenommen worden. Es handelt sich dabei um ein fig-FORTH für den Prozessor U880, um das hier vorgestellte U8000-FORTH sowie um das in /6/ beschriebene Z8000-FORTH. Damit soll einerseits die Kode- und Laufzeiteffizienz des Prozessors U8000 gegenüber einem U880 dokumentiert, andererseits aber auch auf spezielle Aspekte der Realisierung dieser Routinen hingewiesen werden.
Die Routine NEXT realisiert in FORTH das Inkrementieren des IP-Registers und den Ansprung des nächsten FORTH-Wortes (daher NEXT) und ist vergleichbar mit dem Inkrementieren des Programm-Zählers in einer realen CPU.

Tafel 1
Takt- und Speicherbedarf der Routinen NEXT und DOCOL dreier FORTH-Versionen

„FORTH-CPU“	U880 fig	U8000 fig	Z8000-FORTH
	IP=BC	IP=R1	IP=R1
	WA=DE	WA=R2	RSP=R14
	RSP in Memory	RSP=R14	
	Takte	Takte	Takte
	10 JP NEXT		
	:		
NEXT:	:		
LD WA, (IP)	7 LD A,(BC)	8 POP R2,	8 POP R2,
INC IP	6 INC BC	@R1	@R1
	4 LD L,A		
	7 LD A,(BC)		
	6 INC BC		
	4 LD H,A		
LD T,(WA)	7 LD E,(HL)	8 POP R3,	
INC WA	6 INC HL	@R2	
	7 LD D,(HL)		
	4 EX DE,HL		
JP (T)	4 JP (HL)	10 JP @R3	10 JP @R2
Summe:	72 3+11 Byte	26 6 Byte	18 4 Byte
DOCOL:			
PUSH RSP,	16 LD HL,	9 PUSH	9 PUSH
IP	(RSP)	@R14,R1	@R14,R1
	6 DEC HL		
	7 LD (HL),B		
	6 DEC HL		
	7 LD (HL), C		
	16 LD (RSP), HL		
	6 INC DE		
LD IP,WA	4 LD C,E	3 LD R1,R2	8 POP
	4 LD B,D		R1,@R15
JP NEXT	(JP NEXT)	(NEXT)	(NEXT)
Summe:	72 13 Byte	12 4 Byte	17 4 Byte

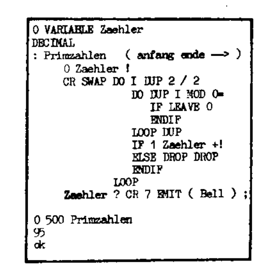
Tafel 2
Laufzeitvergleich für vier FORTH-Versionen (Taktfrequenz jeweils 2,5 MHz)

U880 fig	U8000 fig Version 1	U8000 fig Version 2	Z8000-FORTH
45 s	14 s	2,7 s	2,6 s

Die Routine DOCOL wird in Worten verwendet, die durch Doppelpunkt-Definition entstanden sind (deshalb DOCOLon). Sie kann als Unterprogramm-Aufruf aufgefaßt werden und entspricht damit einem CALL-Befehl bei einer realen CPU.
Die Routine NEXT wird üblicherweise nur einmal im FORTH-Kern vorgesehen und dann jeweils von den entsprechenden maschinenkodepassagen angesprungen (das bedeutet für den Sprung beim U8000 4 Bytes und 7 Takte). In den beiden 16-Bit-FORTH-Versionen wurde eine andere Möglichkeit erprobt, indem diese Routine jeweils an das Ende der entsprechenden Maschinenkodepassagen angehängt wurde und so eine etwas schnellere Abarbeitung erzielt werden konnte, da der Sprung zur NEXT-Routine entfällt. Dadurch verkürzt sich die Programmlaufzeit um etwa 15 Prozent. Beim vorgestellten U8000-FORTH wird dies erkauft durch jeweils 6-Byte-Speicherbedarf pro NEXT-Aufruf gegenüber einem 4-Byte-Sprungbefehl zu einer zentralen NEXT-Routine (d. h. bei etwa 70mal NEXT 140 Bytes mehr).
Einer zentralen NEXT-Routine ist allerdings doch der Vorzug zu geben, da sie es ermöglicht, an definierter Stelle Interrupt-Freigabe und -Sperrung vorzunehmen und für Testzwecke sehr einfach Manipulationen (beispielsweise Arbeit mit Unterbrechungspunkten) zuläßt.
Die Tafel 1 macht die Vorteile des U8000 gegenüber dem U880 sowohl im Hinblick auf Code-Effizienz als auch im Hinblick auf die Laufzeit deutlich. Diese Vorteile sind bei den Routinen NEXT und DOCOL insbesondere der Fähigkeit des U8000 zuzuschreiben, jedes Register (außer R0) als Stackpointer verwenden zu können.
Der Vergleich zwischen den beiden 16-Bit-Versionen läßt erkennen, daß das fig-kompatible FORTH zwar bei NEXT 8 Takte mehr benötigt, dafür aber bei der DOCOL-Routine 5 Takte gespart werden. Wenn auch die NEXT-Routine häufiger aufgerufen wird als DOCOL, so läßt dieser Vergleich doch auf nur geringfügig größere Laufzeiten des fig-kompatiblen FORTH schließen, dafür aber unter Wahrung des fig-Standards.
Ein Test mit dem in Bild 1 aufgeführten Primzahlprogramm (entnommen aus /4/, S. 315, geringfügig geändert), das die Anzahl der Primzahlen in einem vorgegebenen Bereich ermittelt, brachte für die Zählung im Bereich von 0 bis 500 die in Tafel 2 dargestellten Ergebnisse.
Die Tafel zeigt eine deutliche Zeitdifferenz zwischen dem U880-fig-FORTH und den 16-Bit-Versionen.
Bei der Version 1 des U8000-FORTH ist dieser Zeitvorteil allerdings nicht so groß, da bei diesem das Arithmetikkonzept vom 8-Bit-FORTH zunächst übernommen wurde, d. h. Verwendung von nur zwei Assemblerroutinen für Multiplikation und Division. Alle anderen FORTH-Worte, die Multiplikation bzw. Division verwenden, sind als Secondaries (aus Adreßliste bestehende Worte) angelegt.

Die zweite Version verwendet für alle FORTH-Arithmetik-Worte Assemblerroutinen, die unter Nutzung der Multiplikations- bzw. Divisionsbefehle des Prozessors entsprechend schneller in der Abarbeitung sind. Damit kommt die Laufzeit des oben angegebenen Beispielprogramms in die Größenordnung des Z8000-FORTH aus /6/, das ebenfalls Assemblerreoutinen für die Arithmetik beinhaltet.

Bild 1 Primzahlprogramm für Laufzeitvergleich



```
0 VARIABLE Zaehler
DECIMAL
: Primzahlen   ( anfang ende --> )
     0 Zaehler !
     CR SWAP DO I DUP 2 / 2
                 DO DUP I MOD 0=
                    IF LEAVE 0
                    ENDIF
                 LOOP DUP
                 IF 1 Zaehler +!
                 ELSE DROP DROP
                 ENDIF
              LOOP
     Zaehler ? CR 7 EMIT ( Bell ) ;

0 500 Primzahlen
95
ok
```

Konkrete Hardware für die Erprobung des U8000-FORTH ist eine ZRE-UNI.USS8000 /1, 2/ mit Speichererweiterung durch OEM-Speicherbaugruppen 2×16 KByte RAM OPS K3525, seriell gekoppelt mit einem Host-Rechner für die Terminal- und Massenspeicherfunktionen. Die Anpassung an andere U8000-Rechnersysteme ist über die oben erwähnten wenigen FORTH-Schnittstellen oder auf Assemblerebene beispielsweise über System-Calls des U8000 möglich.
Zur Unterstützung der FORTH-Programmentwicklung sind ein leistungsfähiger Screen-Editor, ein Recompiler und Tracer sowie ein 1-Pass-Assembler vorgesehen.

Literatur

/1/ Rehm, W.: Universelle ZRE-Karte ZRE.UNI-USS8000. Dokumentation TU Karl-Marx-Stadt, Sektion Informationstechnik
/2/ Rehm, W.: Universelles 16-bit-System USS8000. Radio, Ferns., Elektron. Berlin 34 (1985) 5, S. 282–285
/3/ Woitzel, E.: comFORTH – Programmierwerkzeug FORTH unter SCP. edv-aspekte, Berlin 5 (1986) 4, S. 47–52
/4/ Zech, R.: Die Programmiersprache FORTH. 2. Auflage, München: Franzis-Verlag 1985
/5/ Kuehnel, C.: Erste Erfahrungen mit FORTH. Radio, Ferns., Elektron. Berlin 35 (1986) 9, S. 553–557
/6/ Odette, L.: Z8000 FORTH. Dr. Dobbs Journal, Nr. 71, Sept. 1982, S. 48–63

✉ KONTAKT ☏

Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Abteilung Rechentechnik und wissenschaftlicher Gerätebau, Warnemünde, 2530; Tel. 57229

Gleitkomma in FORTH

Bodo Bachmann
Friedrich-Schiller-Universität Jena, Sektion Physik

FORTH verfügt in der fig-Standardversion über 2- und 4-Byte-Integerarithmetik. Oft kann man aber auf Gleitkommarechnung und die Verwendung höherer mathematischer Funktionen nicht verzichten. Eine günstige Variante, FORTH für solche Anwendungen zu erweitern, besteht im Einbinden eines in sich geschlossenen Gleitkommakalkulators. Leistungsfähige Kalkulatoren von etwa 1,5 KByte Länge findet man z. B. im Betriebssystem des ZX 81 /1/ oder des ZX Spectrum.
Für die Anwendung durch FORTH sind die Softwareschnittstellen eines solchen Kalkulators wichtig. Die oben genannten Kalkulatoren verwenden einen extra Gleitkommastack und ein Zahlenformat mit 1 Byte Exponent und 4 Byte Mantisse. Der Kalkulator wird mit RST 28 aufgerufen. Es folgt ein Makrocode, der die gewünschte Operation kennzeichnet. Jeder Aufruf wird mit 34H bzw. 38H abgeschlossen. Außerdem sind Unterprogramme zur Umwandlung zwischen 2-Byte-Integer- und 5-Byte-Gleitkommaformat sowie zur Zahlenkonvertierung vorhanden. Auffallend und für die beschriebene Anwendung äußerst günstig ist die Ähnlichkeit der oben genannten Kalkulatoren mit der Programmiersprache FORTH.

Eingabemodus

Das erste Problem ist die Eingabe einer Gleitkommazahl in FORTH. Die Verarbeitung einer Eingabe erfolgt durch das FORTH-Wort INTERPRET. Kann ein eingegebener String nicht als Wort im Dictionary interpretiert werden, so wird versucht, den String als Zahl zu verwerten. Dazu wird das Wort NUMBER verwendet. In /2/ sind INTERPRET und NUMBER ausführlich beschrieben. Doppelt genaue Zahlen mußten bisher einen Punkt beinhalten. Im Bild 1 ist gezeigt, wie man das Wort NUMBER korrigiert, um jetzt doppelt genaue Zahlen mit Komma einzugeben, so daß der Punkt für Gleitkommazahlen reserviert bleibt.

Um jetzt auch Gleitkommazahlen eingeben zu können, sind alle Änderungen bzw. Neudefinitionen entsprechend Bild 2 durchzuführen.

Im Wort INTERPRET wird die Code-Feld-Adresse (CFA) von NUMBER durch die von GNUMBER ersetzt. Damit beschränken sich die Änderungen im FORTH-Kern auf 4 Byte (2 Byte in NUMBER und 2 Byte in INTERPRET). Alles andere sind Neudefinitionen.

Bild 1 Änderung im Wort NUMBER

Bild 2 Änderung im Wort INTERPRET und zusätzliche Definitionen

Bild 3 Definitionsworte für Gleitkommakonstanten und -variablen

Bild 4 Definitionswort zum indirekten Kalkulatoraufruf

Friedrich-Schiller-Universität Jena. Die Spezialisierung erfolgte im Forschungszentrum des VEB Carl Zeiss JENA auf dem Gebiet der Prüfung optischer Systeme.
Das Thema seiner Diplomarbeit lautet „Physikintensive Softwareentwicklung zum Themenkomplex der Wellenflächenanalyse". Dazu hat er sich intensiv mit der Programmiersprache FORTH befaßt und diese mit wesentlichen Ergänzungen wie Gleitkommarechnung, Editor, Relocater und Recompiler auf einem erweiterten robotron Z1013 implementiert.

Die Wörter GLIT, GLITERAL und G, haben die gleiche Aufgabe wie LIT, LITERAL und, (Komma), nur daß sie für 5- und nicht für 2-Byte-Zahlen zuständig sind. Sie sind entsprechend zu definieren.

Das Wort PKONT testet im Wort GNUMBER den Eingabestring auf das Vorhandensein eines Punktes und setzt ein Flag. Ist das Flag *false,* so wird mit NUMBER fortgesetzt, wie es auch ursprünglich war. Ist das Flag *true,* so wird durch KAL die Zahl konvertiert und auf dem Gleitkommastack abgelegt. Mit der Folge R> DROP 51E3 >R ;S wird die normale Return-Adresse durch die Adresse der CFA des Wortes ?STACK in INTERPRET (51E3 steht nur als Beispieladresse) ersetzt, und mit ;S setzt das Programm auch dort fort.

Die Ausgabe einer Gleitkommazahl erfolgt einfach unter Aufruf des Unterprogramms zur Rückkonvertierung im Kalkulator und anschließender Stringausgabe, z. B. mittels TYPE.

Speicherorganisation

Die Befehle G@ und G! (Funktionsweise analog @ und !) kann man leicht als Primitive (Maschinendefinition) unter Verwendung des LDIR-Befehls realisieren.

Für Wörter wie G→S und S→G (5-Byte-2-Byte-Umwandlungen) stehen Unterprogramme im Kalkulator bereit.

Bild 3 zeigt die Definitionsworte zur Vereinbarung von Gleitkommakonstanten und -variablen. Werden nun mittels dieser neuen Definitionsworte Gleitkommakonstanten bzw. -variablen definiert, so wird bei Aufruf einer Gleitkommakonstanten ihr Wert auf den *Gleitkommastack,* bei Aufruf einer Gleitkommavariablen jedoch deren Adresse auf den *Parameterstack* gelegt.

Gleitkommafunktionen

Um die volle Leistungsfähigkeit des Kalkulators auszunutzen, ist es günstig, ein neues Definitionswort aufzustellen (siehe Bild 4). Mit dem Wort GPAK können nun leicht FORTH-Worte definiert werden, die diese Kalkulatorfunktionen aufrufen. Ein Beispiel soll dies verdeutlichen. Im Kalkulator des ZX 81 entspricht der Sinusfunktion der Makrocode 27H. Durch Eingabe der Sequenz 27 GPAK SIN ist das Wort SIN in FORTH definiert. Legt man nun eine Gleitkommazahl auf den Gleitkommastack und ruft SIN auf, so steht danach der Sinuswert auf dem Gleitkommastack (der alte Wert wurde überschrieben)! Folgendes ist dabei abgelaufen: Schon bei der Definition des Wortes SIN

1

```
SCR # 500

 : NUMBER
    0 0 ROT DUP 1+
    C$ 002D = DUP >R + FFFF
    BEGIN
       DPL ! (NUMBER)
       DUP C$ BL -
    WHILE
       DUP C$ 002E        ( 2E durch 2C ersetzen )
       - 0 ?ERROR 0
    REPEAT
    DROP R> IF
              DMINUS
            ENDIF
```

2

```
SCR # 501

 : INTERPRET
    BEGIN -FIND IF STATE $ < IF CFA ,
    ELSE CFA EXECUTE ENDIF ?STACK
    ELSE HERE NUMBER             ( NUMBER wird von GNUMBER ersetzt)
    DPL $ 1+ IF [COMPILE] DLITERAL
    ELSE DROP [COMPILE] LITERAL
    ENDIF ?STACK ENDIF AGAIN ;
```

```
SCR # 502

 CODE PKONT
              POP HL              ( Stringanfangsadresse)
        M1:   LD A,(HL)
              CMP 20H             ( Vergleich mit Delimiter)
              JRZ M2
              INC HL
              CMP 2EH             ( Vergleich mit Punkt)
              JRNZ M1
              LD HL,0001          ( Flag = 1)
              JMP HNEXT
        M2:   LD HL,0000          ( Flag = 0)
              JMP HNEXT           ( Sprung Innerer Interpreter)
 END-CODE
                 >
```

```
SCR # 503

 CODE KAL
              POP DE              ( Stringanfangsadresse)
              PUSH BC
              LD (ELINE),DE       ( in Arbeitszelle d. Kalkulators)
              CALL KONV           ( UP zur Konvertierung)
              POP BC
              JMP NEXT

 END-CODE

 : GNUMBER
     DUP 1+ PKONT IF 1+ KONV
      [COMPILE] GLITERAL
     R> DROP 51E3 >R ;S
     ENDIF NUMBER ;
```

3

```
SCR # 504

 : GCONSTANT
   CREATE SMUDGE G, ;CODE
          INC DE         ( Zeiger auf erstes der 5 Byte)
          PUSH DE
          JMP -> G$      ( weiter wie G$)

 : GVARIABLE
   GCONSTANT ;CODE
          JMP -> VARIABLE ( weiter wie VARIABLE)
```

4

```
SCR # 505

 : GPAK
   CREATE SMUDGE C, ;CODE
     DOGPAK:    INC DE       ( Zeiger auf Makrocode)
                LD A,(DE)
                LD (ADR),A   ( Makrocode auf ADR laden)
                PUSH BC
                RST 28       ( Kalkulatoraufruf)
         ADR:   MC
                MC           ( Makrocode f. Kalkulatorstop)
                POP BC
                JMP NEXT     ( Sprung Innerer Interpreter)
```

wurde auf dessen CFA durch das Wort (;CODE), welches indirekt in GPAK vorkommt, der Inhalt durch die Adresse DOGPAK ersetzt. Bei Aufruf von SIN wurde nun der ab DOGPAK stehende Maschinencode ausgeführt. Hier wurde zuerst der Makrocode auf die Adresse ADR geschrieben und dann der Gleitkommakalkulator aufgerufen.
So sind mit geringstem Speicherbedarf alle Funktionen, die der Kalkulator vorsieht, definierbar.
Mithin sind alle Funktionen, wie sie z. B. ein leistungsfähiges BASIC bietet, auch in FORTH vorhanden. Der wesentlichste Vorteil besteht jedoch in der etwa 10fachen Verarbeitungsgeschwindigkeit von FORTH bei typischen mathematischen Anwendungen, wie Koordinatentransformationen und Gleitkommamatrizenrechnung. Dieser Faktor setzt sich zusammen aus der Tatsache, daß FORTH-Programme nicht nur aus Gleitkommawörtern bestehen und daß die Gleitkommaroutinen von FORTH schneller als von BASIC aufgerufen werden können.
Abschließend noch ein Hinweis zur Fehlerbehandlung. Im Kalkulator wird eine eigene Fehlerauswertung vorgenommen. Wichtig ist, daß bei einem aufgetretenen Fehler, z. B. Division durch Null, wieder zum FORTH zurückgesprungen werden muß. In den beiden ZX-Kalkulatoren wird auf eine Arbeitszelle eine Fehlernummer gelegt und anschließend mit der Adresse, die über dem Initialisierungswert des Stack-Pointers liegt, fortgefahren. Während der Initialisierung von FORTH kann dorthin z. B. die Adresse des Warm-Start-Entry gelegt werden.
Eine solche Gleitkommaversion wurde auf einem robotron Z1013 getestet und in einem Grafikprogramm angewendet, welches in einem folgenden Artikel beschrieben werden soll.

Literatur

/1/ Heß, E.: Schnelles Rechnen mit dem ZX 81. Haar: Markt und Technik 1984
/2/ Zech, R.: Die Programmiersprache FORTH. München: Franzis-Verlag 1984
/3/ Kühnel, C.: Erste Erfahrungen mit FORTH. Radio, Ferns., Elektron. Berlin 35 (1986) 9, S. 553–557

✉ KONTAKT ☏

Forschungszentrum des VEB Carl Zeiss JENA, Abt. WOG 1, Carl-Zeiss-Str. 1, Jena, 6900; Tel. 83 29 63.

Steuerung des Plotters K 6418 in FORTH

Bodo Bachmann
Friedrich-Schiller-Universität Jena, Sektion Physik

Der Digitalplotter robotron K 6418 ist ein Zeichengerät, welches vorwiegend für grafische Zwecke geeignet ist. Die Verbindung zum Mikrorechner erfolgt über eine SIF-1000- oder eine IFSP-Schnittstelle. In der hier beschriebenen Anwendung wurde vom Mikrorechner eine PIO als IFSP-Interface verwendet. Die Datenleitungen wurden an PIO-Port A angeschlossen. Von PIO-Port B wurden Bit 0 mit SC (Sender bereit) und Bit 1 mit AC (Empfänger bereit) verbunden. Die Leitung AO (Betriebsbereitschaft Sender) wurde an Masse gelegt, und die Leitung SO (Betriebsbereitschaft Empfänger) wurde nicht verwendet. Bild 1 zeigt dazu die Initialisierungsroutine sowie die Kernroutine zur Ausgabe von ASCII-Zeichen an den Plotter. Im Registerpaar HL wird die Anfangsadresse des Strings erwartet. Die Ausgabe erfolgt bis zum Vorfinden einer Null.

```
3000 C30630   STB: JMP INI
3003 C31D30        JMP AUS

3006 211730   INI: LD HL,03017H      ( Initialisierungsroutine)
3009 0EFF          LD C,0FFH
300B 0603          LD B,003H
300D EDB3          OTIR
300F 0EDF          LD C,0DFH
3011 0603          LD B,003H
3013 EDB3          OTIR
3015 C9            RET
3016 FF            RST 7
3017 CF            RST 1
3018 00            NOP
3019 07            RLCA
301A CF            RST 1
301B 02            LD (BC),A
301C 07            RLCA

301D DB9F     AUS: IN 09FH
301F CB4F          BIT 1.A           ( Empfaenger bereit ? )
3021 20FA          JRNZ AUS
3023 3EFF          LD A,0FFH
3025 D39F          OUT 09FH          ( Sender nicht bereit )
3027 7E            LD A,(HL)
3028 2F            CPL
3029 D3BF          OUT 0BFH          ( Datenausgabe )
302B 3E00          LD A,000H
302D D39F          OUT 09FH          ( Sender bereit )
302F 23            INC HL
3030 7E            LD A,(HL)
3031 FE00          CMP 000H          ( Delimiter )
3033 C8            RZ
3034 18E7          JR AUS
```

Bild 1 Initialisierungs- und Ausgabe-Kernroutine

Die Abarbeitung der übertragenen Kommandos wird im Plotter durch einen implementierten 8-Bit-Mikrorechner vorgenommen. Der Pufferspeicher von 512 Byte ermöglicht eine hohe Zeichengeschwindigkeit. Um diese voll nutzen zu können, bietet sich die Programmiersprache FORTH mit einer hohen Verarbeitungsgeschwindigkeit an. Dabei müssen die Steuerkommandos für den Plotter folgender Syntax entsprechen: Jedes Kommando besteht aus zwei unmittelbar aufeinanderfolgenden Buchstaben und evtl. nachfolgenden Parametern. Die Parameter werden durch Separatoren getrennt, das Kommando wird mit einem Terminator abgeschlossen.

Für die Anwendung in FORTH ist es günstig, drei Syntaxtypen zu unterscheiden. Einmal Kommandos ohne Parameter, dann Kommandos mit einem und schließlich Kommandos mit zwei Parametern.

Bild 2 FORTH-Wort zur Ausgabe an den Plotter

```
CODE PLT

E1         POP HL
C5         PUSH BC
CD 1D 30   CALL 301D
C1         POP BC
C3 9D 43   JMP NEXT
```

Das Wort PLT (Bild 2) stellt die Einbindung der Kernroutine aus Bild 1 in FORTH dar. Die Anfangsadresse des zu übergebenden Strings wird dabei auf dem Stack erwartet. Kommandos ohne Parameter können als feste Bytefolgen im Speicher abgelegt werden. Das Kommando zum Senken der Feder z. B. heißt PD; und wird als Folge 50H 44H 3BH 00H codiert. Steht diese Folge nach der Kernroutine auf Adresse 3036, so lautet das entsprechende FORTH-Wort:

: PD; 3036 PLT ;

Für Kommandos mit Parametern muß der String immer erst zusammengesetzt werden, bevor er an den Plotter gesendet werden kann. Dabei kann ähnlich verfahren werden wie bei der normalen Zahlenausgabe in FORTH. Im Bild 3 werden dazu 5 Hilfsworte definiert. Zu beachten ist, daß die Zeichenfolge rückwärts aufgebaut wird.
Das Wort TYPO setzt die Null und ein Semikolon als Begrenzer. Danach wird der letzte Parameter konvertiert. TYP3 konvertiert bei zwei notwendigen Parametern die erste Zahl und setzt als Zwischenzeichen ein Komma. TYP4 schließlich setzt die vorher auf den

```
: TYP0
<# 00 HOLD 3B HOLD S->D SWAP OVER
DABS #S SIGN 2DROP ;

: TYP3
2C HOLD S->D SWAP OVER DABS #S
SIGN 2DROP ;

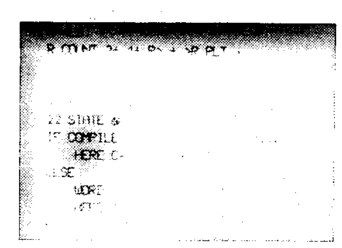
: TYP4
20 HOLD HOLD HOLD H...

: TYP1
ROT TYP0 ...

: TYP2
```

Bild 3 Hilfsworte zum Aufbau des auszugebenden Strings

Bild 4 Direkte Stringausgabe



Stack zu legenden zwei Kommandobuchstaben, stellt die Anfangsadresse des aufgebauten Strings fest und sendet diesen mit PLT an den Plotter.
Als Beispiel für Kommandos mit einem Parameter soll die Definition des Wortes SL (slant instruction) gezeigt werden. Es bewirkt bei Textdarstellungen eine Schrägstellung der Schriftzeichen.

: SL 53 4D TYP1 ;

SL erwartet dann auf dem Stack den Parameter (slant-Winkel). Analog erfolgt die Definition für zweiparametrige Kommandos. Hier das Wort PA (für plot absolute):

: PA 50 41 TYP2 ;

Ein spezielles Wort LB ist im Bild 4 definiert. Es gestattet die Ausgabe eines ASCII-Strings an den Plotter ohne Umweg über die Zahlenkonvertierung. Zum Beispiel bewirkt die Folge

LB LB1234

das Zeichnen der Zahl 1234 auf dem Plotter. Die Folge

LB PA 100 100 ;

hingegen bewegt die Feder des Plotters zu den Koordinaten 100 , 100 .

Der Plotter ist nun vollständig von FORTH aus steuerbar. Dabei ist FORTH selbst bei Programmen mit höherem mathematischen Aufwand, wie z. B. bei einem Programm zur 3dimensionalen Darstellung von Funktionen zweier Veränderlicher, so schnell, daß die Zeichengeschwindigkeit nur durch die Verarbeitungsgeschwindigkeit des Plotters begrenzt ist.
Die Anwendung der Plotterfunktionen in einem Grafikprogramm wird in einem der folgenden Hefte der *Mikroprozessortechnik* gezeigt.

Hardware-Realisierung von FORTH

Gert-Ulrich Vack
Akademie der Wissenschaften der DDR,
Zentralinstitut für Kybernetik
und Informationsprozesse

1. Grundlagen

Bei der Anwendung von FORTH wirkte sich in einigen Fällen bisher die gegenüber reiner Assemblerprogrammierung niedrigere Laufzeiteffektivität aus; in der Literatur werden für den Geschwindigkeitsverlust Faktoren von 1,2 bis 10 (im Vergleich zu reinen Assemblerprogrammen) und 2 bis 4 (bezüglich der Programmierung in C oder FORTRAN) ausgewiesen. (Dennoch gehört FORTH zu den schnellen Programmiersprachen!) Bereits in den siebziger Jahren wurde deshalb mit „reinen" Hardware-Lösungen von FORTH experimentiert. Unabhängig von FORTH wird seit längerem eine Diskussion zu RISC- (Reduced Instruction Set Computer) Architekturen geführt. Bei diesem Konzept wird die Architektur bewußt einfach und orthogonal gehalten, womit VLSI-Prozessoren großer Verarbeitungsbreite effektiv realisierbar sind. Der „Verlust" an Leistung gegenüber CISC (Complex Instruction Set Computer)-Architekturen (dazu gehören die heute bekannten 16- und 32-Bit-Universalprozessoren) wird durch mächtigere Compiler und Betriebssysteme ausgeglichen.
Mit jüngsten Hardware-Entwicklungen wird der Beweis angetreten, daß FORTH-Prozessoren im Sinne von RISC-Architekturen implementierbar sind (unteres VLSI-Niveau) und eine hohe Systemleistung ermöglichen. Daneben werden in Universalprozessoren (z. B. Z8800 /1/) Maßnahmen zur Hardware-Unterstützung von FORTH-ähnlichen Sprachen vorgesehen (einfache Realisierungsmöglichkeiten für den Adreßinterpreter als zeitkritischsten Teil eines FORTH-Systems).
Die Relevanz eines FORTH-Prozessors liegt in der flexiblen Einsetzbarkeit für unterschiedlichste Anwendungsklassen, bevorzugt in den Bereichen der Kleinautomatisierung, der Meßtechnik, sowie in flexibel an neue Einsatzbedingungen anpaßbaren gerätetechnischen Lösungen (etwa intelligente Sensoren, Prozeßsteuerung, Antriebstechnik und damit z. B. das weite Feld der Robotertechnik, der Werkzeugmaschinen oder der flexiblen Fertigungssysteme).
Im folgenden soll deshalb ein Überblick zu den international bekannt gewordenen Lösungen gegeben werden.

2. FORTH-Karte mit 6502

Einer der ersten Versuche in Richtung FORTH-Prozessor war die Essex Forth Microcard aus dem Essex Electronics Centre of the University of Essex. Auf einer 80 mm mal 100 mm großen Leiterplatte wird mit 7 IS auf Basis des Rockwell-Prozessors R 6502 (hier zum FORTH-Prozessor R 65F12 modifiziert) ein FORTH-System im ROM angeboten, welches bei 1 MHz Taktfrequenz betrieben wird. Der Speicherraum beträgt 16 KByte. 40 E-/A-Leitungen werden getrieben. Über RS 232 ist ein Terminal anschließbar. Der RSC-FORTH-Kernel belegt 3 KByte. 10 Interruptquellen und 2 programmierbare 16-Bit-Zähler/Timer können für Echtzeitaufgaben genutzt werden. Mit Einsatz des 65-FRI-Development-ROM wird die Karte zum Entwicklungssystem erweitert. Ein Anwendungsbeispiel ist ein automatischer Tester für die Strom-/Spannungskalibrierung (6 Analogeingänge, 3 zusätzliche Schaltkreise einschließlich 8-Kanal-AD-Umsetzer). Das Programm (Umfang eine A4-Seite Quelltext, 15 FORTH-Worte) realisiert eine Toleranzfeldprüfung jedes Einganges und zeigt optisch die erforderlichen Abgleicheingriffe an.

3. MF16LP-Single Board Language Processor

Die britische Firma Metaforth Computer Systems Limited entwickelte im Doppeleuropaformat eine Leiterkarte mit Bipolar-Kundenwunschschaltkreisen und HMOS-RAM zur Hardware-Realisierung von FORTH. Der Rechner hat eine Verarbeitungsbreite von 16 Bit sowie 24 Adreßleitungen. Für Stacks und Register werden 2 KByte High-Speed-RAM verwendet.
Der Prozessor wird mikroprogrammiert und realisiert etwa 60 FORTH-Worte. Bei Verwendung von Speichern mit 60 ns Zugriffszeit werden die meisten FORTH-Operationen in 100 bis 550 ns abgearbeitet; das entspricht etwa 2 bis 10 Millionen FORTH-Befehlen je Sekunde. Ein solches FORTH-System ist damit etwa 50- bis 100mal schneller als eine Software-Emulation von FORTH auf dem Z 80. Die Nestungsoperationen werden im Zeitschatten der Befehlsabarbeitung realisiert, so daß kein Interpretationsoverhead für die Unterprogramm- bzw. Fadencodetechnik entsteht. Auf dem System sind Multitasking-Versionen von FORTH 79 und FORTH 83 sowie ein C-Compiler einsetzbar. Außerdem ist auch PASCAL verfügbar. Module,die in verschiedenen Sprachen programmiert wurden, können gemixt und gemeinsam abgearbeitet werden.
Zur Systementwicklung sowie -realisierung werden eine CPU-Karte, ein Entwicklungsmodul für IBM-PC sowie ein VME-Bus-Entwicklungssystem angeboten. Damit sind die Anwendungslösungen (Echtzeitsysteme für Bildverarbeitung, Grafik, CAD/CAM oder Expertsysteme usw.) zu implementieren. Für Multiprozessorkonfigurationen bestehen Kopplungsmöglichkeiten über 6 Hochgeschwindigkeits-E-/A-Kanäle mit Übertragungsraten von 10 MBit/s.
In /2/ werden Ergebnisse von Laufzeituntersuchungen angegeben. Der Eratosthenes Sieve (ein Standard-Benchmark-Programm, bei dem die Primzahlen zwischen 3 und 16381 zehnmal gezählt werden) benötigt danach für den MF16LP in FORTH 1,09 s, auf dem 4-MHz-Z80 76 s, auf dem 5-MHz-8086 64 s und auf einer PDP 11/70 11,8 s. Eine C-Realisierung dieses Algorithmus auf der PDP 11/70 benötigt 1,52 s.
Mit aller Vorsicht läßt sich abschätzen, daß ein FORTH-Prozessor mit einer solchen Verarbeitungsleistung eine Reihe von (nicht zu anspruchsvollen) Signalverarbeitungsaufgaben lösen kann.
Eine Besonderheit des MF16LP ist die Metaforth Soft Architecture. Dabei wird der Anwender in die Lage versetzt, nach der Austestung eines Programmsystems mit dem Metaforth Profiler zeitkritische Stellen im Programmlauf zu identifizieren. Spezialisten der Herstellerfirma können anschließend Ergänzungen im Mikroprogramm des Rechners vornehmen, um eine höhere Abarbeitungsgeschwindigkeit der fraglichen Programmteile zu erzielen. Mit diesem Konzept wird eine Brücke geschlagen zwischen der Starrheit von universellen, komplexen Mikroprozessorarchitekturen und der mit wesentlich höheren Kosten für den Anwender erkauften Flexibilität von Bit-Slice-Realisierungen eines Spezialprozessors. Die Softwarekonzeption gestattet es, einen Spezialprozessor in der Firmware des FORTH-Prozessors zu realisieren.

4. FORTH-Prozessor NC 4000

Der markanteste Schritt in Richtung FORTH-Prozessor war die Entwicklung des NC 4000

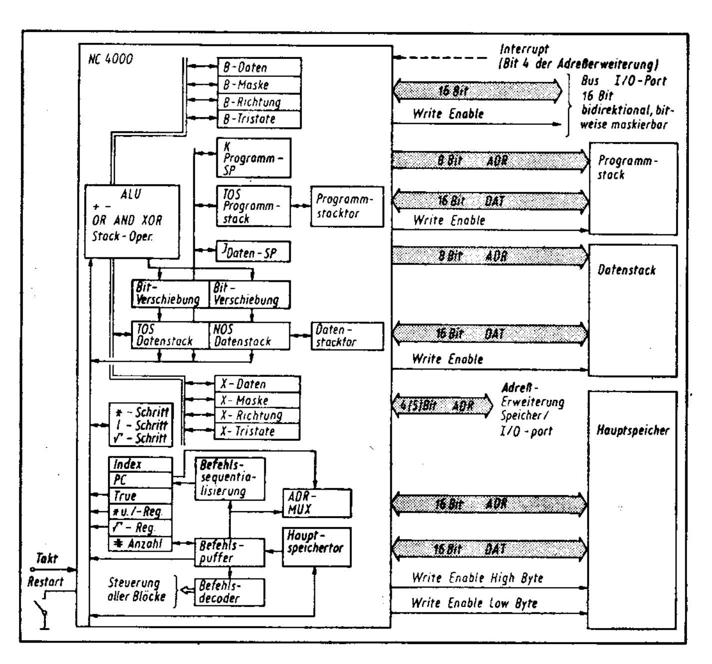


Bild 1 Interne Struktur des NC 4000 (aus der verbalen Beschreibung in /5/ abgeleitet)
TOS – Top Of Stack ≙
1. Element
NOS – Next Over TOS
≙ 2. Element

durch Cupertino/Kalifornien gemeinsam mit dem Begründer von FORTH, Charles Moore. Inzwischen sind bereits verschiedene Realisierungsvarianten dieses Prozessors bekannt geworden.

Der NC 4000 (Bild 1) ist ein 16-Bit-Prozessor und arbeitet voll statisch mit maximal 8 MHz Taktfrequenz. Der Chip ist in einem 120poligen Gehäuse untergebracht. Erste Muster wurden in HCMOS-Gate-Array-Technik gefertigt; intern ist der Schaltkreis überwiegend in Bit-Slice-Technik ausgeführt. Der Prozessor hat keine Mikroprogrammsteuerung und ist deshalb in der Firmware nicht erweiterbar. Daraus ergeben sich jedoch Geschwindigkeitsvorteile. Die meisten der 40 implementierten FORTH-Primitiven werden in einem Takt ausgeführt (einschließlich Anweisungen für strukturierte Programmierung). Zusätzlich können auch 130 Kombinationen aus Primitiven in einem Wort codiert und somit in einem Takt ausgeführt werden. Daraus resultiert eine enorme Verarbeitungsleistung, die über der des M 68000 liegen soll /2/.

In den Befehlscodes wird das oberste Bit ausgewertet. Ist es auf Eins gesetzt, so bezeichnen die nachfolgenden Bits die Adresse eines anderen Wortes innerhalb des Fadencodes (Secondary, Nestungsoperation). Ist das Bit 0, so handelt es sich um den Code einer sofort ausführbaren Primitive. Ein weiteres Bit kennzeichnet in den Befehlscodierungen eine Rückkehranweisung in die aufrufende Ebene. Dieses „Return" wird parallel zur Verarbeitungsoperation ausgeführt, so daß (ebenso wie beim MF16LP) der Interpretationsoverhead entfällt.

Die Reduzierung des Adreßraumes durch die vordefinierten Bits in den Befehlscodes ist bedeutungslos, da der generierte Code außerordentlich kompakt ist. Der gesamte adressierbare Speicher umfaßt 64 KWorte. Mit Nutzung der Erweiterungsleitungen kann er bis auf 2 MByte expandiert werden. Daten- und Programmstacks werden als getrennte Speicherblöcke realisiert und über getrennte Busse erreicht. Das ermöglicht eine extrem kurze Umschaltzeit bei Context-Switch sowie eine Parallelarbeit bei Hauptspeicher- und Stackzugriffen. Die oberste der 257 Programmstackzellen sowie die obersten 2 der

Dipl.-Ing. Gert-Ulrich Vack studierte von 1972 bis 1976 an der Technischen Universität Dresden Informationstechnik. Nach Erlangung des Diploms begann er seine Tätigkeit als wissenschaftlicher Mitarbeiter im ZKI der AdW der DDR. Er beschäftigte sich mit dem Aufbau, der Programmierung und der Anwendung von Mikrorechnern für industrielle Steuerungsaufgaben. Außerdem befaßte er sich mit Architekturen von modernen Mikroprozessoren und mit dem objektorientierten Softwareentwurf. Seit 1981 arbeitet er intensiv an der Implementierung, Nutzung und Verbreitung von FORTH-Systemen.

258 Datenstackzellen liegen im Schaltkreis als Register vor. 16 E-/A-Kanäle können bidirektional, maskierbar, selbstvergleichend und für gelatchte bzw. Tristate-Ausgaben genutzt werden. Gegebenenfalls können auch die 5 Erweiterungsleitungen für die Expandierung des Speicheradreßraumes als E-/A-Leitungen betrieben werden. Eine dieser Leitungen ist darüber hinaus auch als Interrupteingang nutzbar.

Der Prozessor greift auf die Stacks, auf den Hauptspeicher und auf die E-/A-Adreßräume parallel während der Verarbeitungsoperationen zu.

Auf Basis des NC 4000 wird ein Einplatinenrechner unter der Bezeichnung SBC Beta-Board als Zusatzkarte für den IBM-PC angeboten. 28 KWorte RAM und 4 KWorte ROM sowie 8 Stacksegmente (Umschaltung in weniger als 5 µs) stehen zur Verfügung, um ein Programmentwicklungssystem auf der Basis von PolyFORTH zu unterstützen. Das Delta Evaluation System SC 10 000 Delta Board ist ein Entwicklungsmodul (4 MHz, 8 Stacksegmente, je 4 KWorte RAM und EPROM, RS-232-Schnittstelle, 21 E-/A-Kanäle). Ergänzungsmoduln (Bus, Peripherie, Speicher) sind angekündigt.

5. Schlußfolgerungen

Aus Untersuchungen der Universität von Hull /3/ geht hervor, daß prozedurale Programmiersprachen auf der Grundlage von FORTH-Architekturen implementierbar sind. Der FORTH-Prozessor könnte dabei als Zielhardware für eine Vielzahl von Programmsystemen sowie Fachsprachsystemen genutzt werden.

Mit den Möglichkeiten der VLSI-Technologie sind FORTH-Prozessoren als Chip realisierbar. Auf Grund der Flexibilität der Software (Implementierungsprinzip, Erweiterbarkeit des Sprachkonzeptes) wird auch die Hardware universeller einsetzbar, womit große Stückzahlen bei der Herstellung des Prozessors ökonomisch realisierbar sind. Die Leistungsmerkmale erreichen oder überschreiten die von 16-Bit-Prozessoren. Selbst für komplizierte Anwendungsfälle (Bildverabeitung, Künstliche Intelligenz) ergeben sich mit dem FORTH-Prozessor interessante Realisierungsalternativen gegenüber Universalprozessoren, wobei einfachere, zuverlässigere, leistungsfähigere und insgesamt ökonomischere Lösungen zu erwarten sind.

Bei der Konzipierung eines FORTH-Prozessors sind immer folgende Aspekte zu berücksichtigen:

– Realisierung als fest verdrahtete oder mikroprogrammierbare Hardware
– Auswahl des Basisbefehlssatzes
– Eignung für den Einsatz in Multiprozessor- und Mehrrechnerkonfigurationen (auch in heterogenen Systemen, gemeinsam mit Universal-, Spezial- und Koprozessoren)
– Auswahl einer geeigneten Programmierumgebung (FORTH 83, Poly-FORTH usw.).

Literatur

/1/ Meininger, W.: „Super 8": Microcontroller mit 8-Bit-Architektur und Hochsprachunterstützung. DESIGN & ELEKTRONIK 5/86, S. 94–99
/2/ FORTH-Systeme für die Industrie. Katalog der Firma Angelika Flesch, Titisee-Neustadt, 1986, S. 27–43
/3/ Kundeninformation zur Interkama 1986
/4/ Strass, H.; Brodie, L.: Der FORTH-Mikroprozessor: Programmiersprache und CPU aufeinander optimal abgestimmt. DESIGN & ELEKTRONIK 5/86, S. 94–99
/5/ NOVIX NC 4000 Microprocessor. Firmenmaterial von Novix, Cupertino
/6/ forth microcard incl. application note. Firmenmaterial von Essex Electronics Centre, Univ. of Essex, Colchester, Essex

🖂 KONTAKT ☎

Akademie der Wissenschaften der DDR, Zentralinstitut für Kybernetik und Informationsprozesse, Kurstr. 33, PF 1298, Berlin, 1086; Tel. Dresden 4633255

Termine

V. Symposium Optoelektronik

WER? VEB Werk für Fernsehelektronik und Betriebssektion der KDT
WANN? 10. bis 12. Dezember 1987
WO? Berlin
WAS?
• Erfahrungsaustausch mit den Kooperationspartnern
• Vorträge und Posterdiskussionen u. a. zu optoelektronischen Sendern, Empfängern und Kopplern, Bauelemente der Lichtleiternachrichtenübertragung
• umfangreiche Ausstellung von Erzeugnissen einschließlich ihrer Anwendung
WIE? Anmeldungen bis zum 15. 9. 1987 an das Werk für Fernsehelektronik, Betriebssektion der KDT, Ostendstraße 1–14, Berlin, 1160.

Dr. Hornung

EMR-Controller für eine LCD-Punktmatrix

Rudolf Lösel
VEB Robotron-Optima Büromaschinenwerk Erfurt

Einführung

In mit Mikroprozessoren gesteuerten Geräten, bei denen eine alphanumerische und/oder grafische Information ausgegeben werden soll, werden immer häufiger Flüssigkristallmatrixanzeigen verwendet. Diese Anzeigen, auch LCD-Module genannt, haben den Vorteil, daß die Darstellungsform und -größe von Zeichen variabel gestaltet werden kann.
Eine Grenze besteht lediglich in der Menge und Aufteilung der Matrixpunkte. Vorteilhaft bei kleineren Geräten ist der niedrigere Strombedarf. Neben einer Betriebsspannung von +5 V ist in einigen Fällen eine zweite Betriebsspannung von −5 V notwendig, die jedoch bei dem geringen Strombedarf, der unter 1 mA liegt, von einem einfachen +5 V/−5 V-Transverter bereitgestellt werden kann.
Die schaltungstechnische Anordnung des LCD-Moduls (Bild 1) ist so gestaltet, daß zwischen dem eigentlichen LCD-Modul und dem Rechner-Anschluß (Bus) ein Controller geschaltet ist, der die notwendige Signal- und Steueranpassung vornimmt.
Der Controller hat folgende Aufgaben zu erfüllen:

- Zwischenspeicherung der anzuzeigenden Daten
- Bildung des anzuzeigenden Punktrasters
- Parallel-Serien-Wandlung des Punktrasters
- Generierung der Steuersignale (meist 4).

Die auf dem LCD-Modul befindlichen Ansteuerschaltkreise U 714 PC haben die Aufgabe, die für die LCD-Matrix spezifischen Signale während der Zeilenanzeigezeit bereitzustellen und zwischenzeitlich die Daten für die nächste Zeile vom Controller zu übernehmen. Durch die Ausstattung des U 714 PC mit seriellem Übernahmeregister und parallelen Ausgaberegistern mit nachgeschalteten Treibern ist es möglich, die Menge der (n + m) Anschlüsse, die bei kleinen Anzeigen bereits 100 überschreitet, auf eine Datenleitung und vier Steuerleitungen zu reduzieren. Auf die Funktion und die Wirkungsweise des U 714 PC wird in diesem Beitrag nicht eingegangen, eine ausführliche Funktionsbeschreibung ist in /1/ enthalten.

Ansteuerung von Anzeige-Matrizen

Die LCD-Module erfordern ein spezielles Steuerprogramm, das entsprechend der Größe der Anzeige modifiziert wird. Neben den seriell bereitzustellenden Anzeige-Daten sind der Ansteuerung Taktsignale zum

- Übernehmen der Daten (CL2)
- Zeilenwechsel (CL1) und
- Bildwechsel (FLM)

zu liefern. Zur Verhinderung von Gleichspannungskomponenten auf dem LCD-Modul ist außerdem ein Wechselsignal (M) im Zeilentakt, bei größeren Anzeigen im Bildwechseltakt, erforderlich. Bild 2 zeigt die Steuersignale für den LCD-Modul MF 01 RW des VEB Werk für Fernsehelektronik für eine Zeilenanzeigezeit. Diese Anzeige besitzt 10 Zeilen mit 120 Spalten. Die mit der fallenden Flanke von CL2 eingeschobene Information wird mit der fallenden Flanke von CL1 zur Anzeige abgespeichert. Die Zeileninformation (Beginn der 1. Zeile) wird bei FLM = H mit der steigenden Flanke von CL1 eingeschoben und mit der ersten steigenden Flanke von CL2 abgespeichert.
Auf den internen Steuerungsablauf zwischen den Ansteuerschaltkreisen und dem Flüssigkristall wird nicht eingegangen, das Prinzip der Multiplexansteuerung von Matrixanzeigen wird ausführlich in /2/ beschrieben. Bei der angegebenen Anzeige ist der Multiplexgrad n = 10. Die Spalten bzw. Zeilen werden mit sechs verschiedenen Spannungspegeln, welche von einem Spannungsteiler auf der Matrix geliefert werden, angesteuert.

Betriebsarten von LCD-Controllern

Der Controller kann in zwei Betriebsarten arbeiten. Für alphanumerische Anzeigen genügt vom Rechner die Übergabe von Anzeigedaten und Anzeigeposition in einen Wiederholspeicher des Controllers, aus dessen Inhalt die Datensignale für die Anzeige gebildet werden. Hierzu ist noch ein interner oder externer Zeichengenerator notwendig. Diese Betriebsart wird Zeichenmode genannt.
Sollen besondere Zeichen und/oder Grafik ausgegeben werden, muß für jeden Punkt der Anzeige die entsprechende Hell/Dunkel-Information durch ein Grafik-Programm gebildet und an den Controller übergeben werden. Diese Betriebsart, Grafik-Mode genannt, benötigt nur einen Wiederholspeicher, der jedoch die gesamte Punktmenge zwischenspeichern muß.

Controllerlösung mit EMR

Im folgenden soll ein Controller in Zeichenmode auf Basis des EMR UB 8810 D angegeben werden, der für die Ansteuerung der LCD-Matrix von 10 × 120 Punkten Verwendung findet. Durch Veränderung von Konstanten des Programms ist eine einfache Umstellung auf Anzeigen anderer Matrixteilungen möglich. Bei der Lösung wurde eine minimale Umgebungsbeschaltung des EMR angestrebt. Die Lösung mit dem EMR wurde gewählt, weil Lösungen mit integrierten Schaltungen des üblichen Sortiments zu umfangreich wären. Von Herstellern der LCD-Matrizen im Ausland werden hierzu spezielle LSI-Schaltkreise angeboten, welche in einigen Fällen mit externem RAM und Zeichengenerator ergänzt werden müssen /3/.

Schnittstelle zum Rechner

Für die Schnittstelle zum Rechner wird die parallele Datenübernahme gefordert. Durch Veränderung des Übernahmeprogramms ist auch eine serielle Schnittstelle möglich.
Der EMR wird dabei mit dem Port 1 direkt an den Datenbus geschaltet. Da der Controller außer den Daten noch Adressen und Befehle übernehmen muß, werden zwei Steuerleitungen /SW (Steuerwort) und /DAT (Daten) benötigt, die bei Daten- oder Befehlsübergabe aktiviert werden. Dadurch werden im Rechner zwei Ausgabeadressen belegt. Um Datenverlust durch Überschreiben der gepufferten Eingangsdaten zu vermeiden, ist entweder Handshakebetrieb notwendig oder die Datenübergabe muß in einem bestimmten Zeittakt erfolgen. Die Daten, Adressen oder Befehle werden an Port 1 angelegt und /DAV1 an P33 generiert. Bei Befehlen oder Adressen erscheint gleichzeitig noch ein Signal an P30, welches einen zweiten Interrupt erzeugt.
Durch seine Abfrage am Anfang der Eingaberoutine kann zwischen Daten und Adressen oder Befehlen unterschieden werden. Außer Daten können folgende Informationen an den Controller gegeben werden:

- Adresse (Anzeigeposition des nachfolgenden Zeichens (Cursoradresse)
- Adresse des Zeichens, welches blinken soll (Blinkadresse)

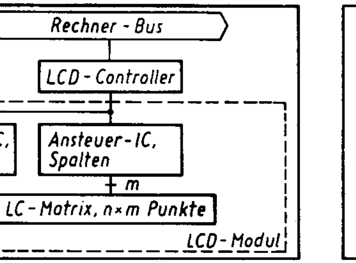


Bild 1 Anschaltung der Anzeige an den Rechner

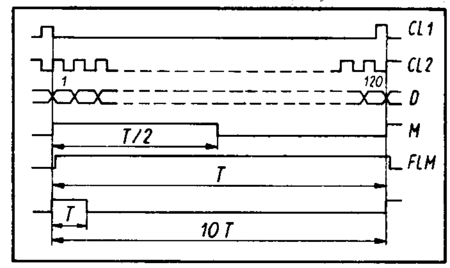


Bild 2 Ansteuersignale für eine Punktmatrix-Anzeige

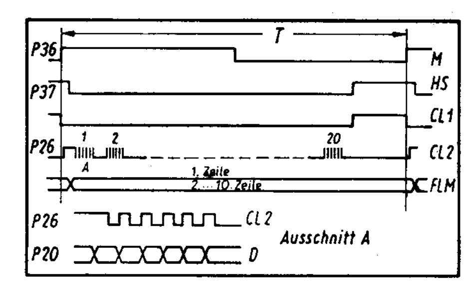


Bild 3 EMR-Steuersignale für die Punktmatrix-Anzeige

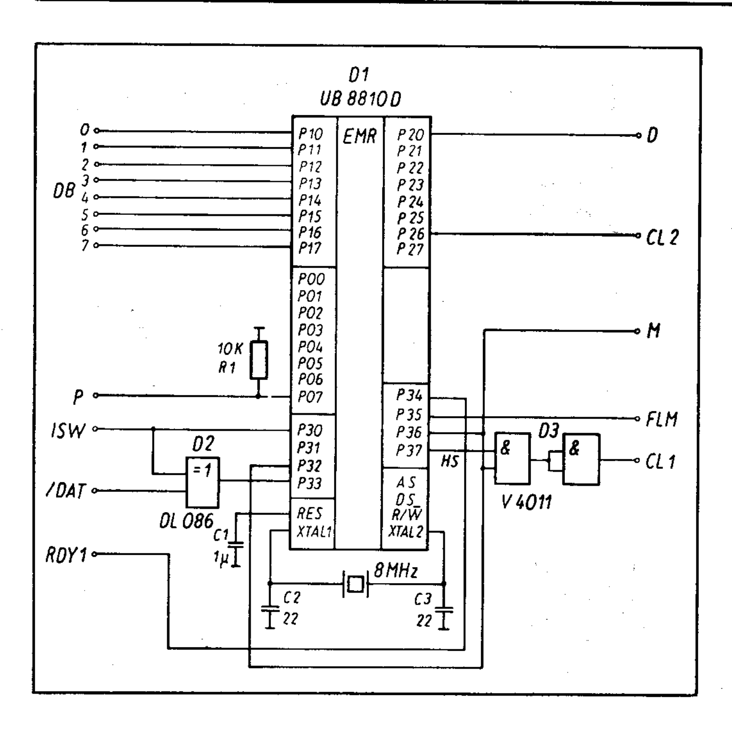


Bild 4 Schaltung des EMR-Controllers

- Blinkadresse löschen
- Anzeige ausschalten (Dunkeltastung)
- Dunkeltastung aufheben (Anzeige einschalten).

Da für die Codierung der Blink- bzw. Cursoradresse maximal 5 Bit benötigt werden, werden die restlichen 3 Bit für die Unterscheidung der Adressen- bzw. Befehlsart genutzt. Die Codierung der verschiedenen Adressen- und Befehlsarten zeigt Tafel 1.
Um die Datenübernahme zu beschleunigen, wird die Adresse des eingegebenen Zeichens nach der Eingabe automatisch um 1 erhöht, so daß die Cursoradresse nur eingegeben werden muß, wenn keine kontinuierliche Datenfolge vom Rechner ausgegeben wird. Die eingegebenen Zeichen werden im Anzeigewiederholspeicher (interner RAM des EMR) abgelegt. Das Eingabeprogramm besitzt gegenüber dem Anzeigeprogramm die Priorität 2.
Bei zeitlich kurzen Eingabeprogrammen kann dem Eingabeprogramm auch die Priorität 1 gegeben werden.

Generierung der Signale für die LCD-Anzeige

Die Matrixanzeige benötigt die im Bild 2 angegebenen Steuer- und Datensignale. Um die Zeitbedingungen für M einzuhalten, wird das Signal vom Timer T1 erzeugt und über P36 ausgegeben. Durch die Rückführung von M auf P32 wird das Zeilenprogramm synchron zu M gestartet. In diesem Programm wird ein Hilfssignal (Programm läuft) an P37 ausgegeben, welches durch logische Verknüpfung mit M CL1 generiert. Das Bildwechselsignal FLM, welches nur während des Wechsels von der 1. zur 2. Zeile H sein muß, wird softwaremäßig erzeugt und an P35 ausgegeben. Das auszugebene Bitmuster für ein 5 × 7- oder 5 × 9-Zeichen wird dem Zeichengenerator entnommen. Die Adresse des Zeichengenerators wird ähnlich wie bei der Bildschirmausgabe aus Zeichen- und Zeilenadresse berechnet. Vor der Parallel-Serienwandlung wird das 5 Bit lange Zeichenmuster b0 bis b4 mit L auf b5 (Zwischenraum) und H auf b6 und b7 (Taktbit) ergänzt und in P2 geladen. Die Daten- und Taktbereitstellung CL2 geschieht durch folgendes Programm (siehe auch /4/):

LDC R2, @PR4	Bitmuster laden (b0 in P20)
AND R2, #BFH	Bit 6 löschen (Takt 1)
SRA R2	Rechtsverschiebung und Setzen von b6 (b1 in P20)
AND R2, #BFH	Takt 2 usw.

Die seriellen Daten D werden an P20 und der Takt CL2 an P26 ausgegeben. Nach fünfmaliger Rechtsverschiebung und sechsmaligem Löschen von b6 sind die 5 Bit und der Zwischenraum ausgegeben. Hierbei werden entgegen den Angaben in Bild 2 die Daten mit ihren Taktimpulsen asynchron bereitgestellt. Die Ansteuerschaltkreise lassen diese Betriebsart zu, Bedingung ist nur, daß die Datenausgabezeit an den Modul kleiner T ist und die Zeitbedingungen zwischen D und CL2 eingehalten werden.
Das Ausgabediagramm zeigt Bild 3. An Sonderfunktionen ist vorgesehen, daß durch einen Cursor in der 9. Zeile die Position angezeigt wird, auf die das nächste Zeichen eingegeben wird; weiterhin kann ein ausgewähltes Zeichen mit einer Frequenz von ca. 2 Hz blinken. Durch Programm läßt sich die Anzeige aus- und einschalten. Nach Anlegen der Betriebsspannung oder nach RESET wird bei der Initialisierung folgender Grundzustand eingestellt:
- Bildwiederholspeicher gelöscht
- Cursor auf Position 1
- Anzeige eingeschaltet
- Blinkadresse ausgeschaltet.

Dipl.-Ing. Rudolf Lösel (62) erwarb an der Technischen Universität Dresden 1962 im Fernstudium das Diplom auf dem Gebiet der digitalen Meßtechnik. Bis 1964 arbeitete er in der Meßgeräteentwicklung des VEB Funkwerk Erfurt. Seitdem ist er im VEB Kombinat Robotron – ab 1982 im VEB Robotron-Optima Büromaschinenwerk Erfurt – im Bereich Forschung/Entwicklung in mittleren Leitungsfunktionen und als Bauelementekonsultant tätig.

Tafel 1 Codierungsarten

Code	Befehl
000CCCCC	Cursoradresse auf 0 ... 19
001BBBBB	Blinkadresse auf 0 ... 19
01000000	Anzeige ein
10000000	Anzeige aus
00111111	Blinkadresse aus

Schaltung des Controllers

Die Gesamtschaltung des Controllers einschließlich seiner Hardware-Ergänzungen zeigt Bild 4. Statt des UB 8810D wurde in der Erprobung mit dem UB 8820D und dem EPROM U 2716C gearbeitet.
Das Programm einschließlich des Zeichengenerators für 128 Zeichen im 5 × 9-Raster ist im ROM bzw. EPROM untergebracht.
Zum Test der Anzeige ist im Programm eine Prüfroutine vorhanden, welche durch Anlegen von +5 V an PO7 aufgerufen wird. Damit ist ein Test des Anzeigemoduls ohne Rechner möglich.

Erprobung

Die Erprobung erfolgte mit einer 16stelligen 5 × 7-Punkt-Anzeige mit Cursor Hitachi H 2532A und einer Anzeige von 10 × 120 Punkten (MF 01 RW), welche die Darstellung von 20 Zeichen im 5 × 9-Raster einschließlich Cursor zuläßt. Die Bildwechselfrequenz betrug dabei 64 Hz (T = 1,56 ms für eine Zeile). Die Generierung der Daten einschließlich Parallel-Serienwandlung benötigt ca. 50 % der zur Verfügung stehenden Zeit T. Damit ist abzuschätzen, daß mit einem EMR bei 8 MHz Quarzfrequenz auch größere Anzeigen bis zu einer maximalen Punktzahl von 2000 Punkten angesteuert werden können.
Bei noch größeren Anzeigen muß entweder die Bildwechselfrequenz zur Vergrößerung von T abgesenkt oder die zeitaufwendige P/S-Wandlung muß hardwareseitig vorgenommen werden. Die Arbeiten wurden durchgeführt, um die Eignung des EMR als Controller für den zu erwartenden Einsatz von LCD-Anzeigen in elektronischen Schreibmaschinen zu überprüfen.

Literatur
/1/ TGL-E 38519, 11/86
/2/ Klein, W.: Ansteuerverfahren für LCDs. Radio, Ferns., Elektron. 35 (1986) 9, S. 590
/3/ Hitachi MOS LSI Data Book, LCD Driver LSI, 1983
/4/ Technische Beschreibung UB 8810 D, UB 8811 D, UB 8820 M, UB 8821 M. VEB Mikroelektronik „Karl Marx" Erfurt

⊠ KONTAKT ☏
VEB Robotron-Optima Büromaschinenwerk Erfurt, Bereich Forschung/Entwicklung, Mainzerhofplatz 13, Erfurt, 5020; Tel. 52 72 41.

Preiswertes Prüfgerät

Frank March, Armin Zaspel, Harald Gürth
VEB Mikroelektronik „Karl Marx" Erfurt

Es wird ein universeller, rechnergesteuerter Leiterplattenprüfaufbau mit sehr flexiblen Einsatzmerkmalen vorgestellt. Dieser unterscheidet sich deutlich von bisher bekannten Lösungen (z. B. automatische Leiterplattentester) durch den äußerst geringen Hardwareaufwand.
Mittels entsprechender Adaptierung können Prüflinge im Nichtechtzeitbetrieb Funktionstests unterzogen werden. MACRO-Befehle erleichtern die Programmierung, spezielle Betriebsarten unterstützen die Fehlerdiagnose. Aufbau und Inbetriebnahme der Lösungsvariante ist unkritisch.
Der Prüfplatz besteht in der Konfiguration aus einem Steuerteil mit Rechner, der auch für andere Steueraufgaben als die Leiterplattenprüfung eingesetzt werden kann, und einem Prüfgerät, das Spannungsquellen, I/O-Tore sowie Steckplätze für Adapter zur Aufnahme der Prüflinge enthält.

Display-Keyboard-Terminal (DKT)
1. Hardware

Das DKT beinhaltet folgende Funktionsgruppen:
- Steuerrechner (U 880, 8K ROM, 4K ROM)
- Display (16-Stellen-7-Segment-Anzeige und je eine LED)
- Tastatur (24 Tasten)
- IFSS-Schnittstelle
- Schnittstelle für Magnetbandgerät (NF-Buchse)
- Parallelschnittstelle (DKT-Bus)
- Netzteil.

Außer Netztrafo, Netzschalter, Netzsicherung, Gleichrichter und Ladekondensator befinden sich alle Bauelemente auf einer Leiterplatte von 22 cm × 33 cm.
Der ROM-Bereich umfaßt 4 Steckplätze für U 2716. Die Ansteuerung des Displays und das Einlegen der Tastatur erfolgt durch Software-Multiplex. Die IFSS-Schnittstelle kann sowohl sender- als auch empfängerseitig durch Einlöten entsprechender Drahtbrücken aktiv bzw. passiv betrieben werden.
Der DKT-Bus ist eine Parallelschnittstelle mit 8 bidirektionalen Daten- und 7 Adreßleitungen. Die Steuersignale sind so gewählt, daß U 855, U 856, U 857 und K-1520-IN/OUT-Leiterplatten ohne Einschränkung betrieben werden können. Weiterhin stehen der NMI-Eingang sowie Eingänge und Ausgang der CTC-Kanäle 2 und 3 zur Verfügung. Die RESET-Leitung läßt sich auch softwaremäßig aktivieren. Vom Ladeelko ist eine Leitung zum 39poligen DKT-Bus-Steckverbinder geführt, so daß sowohl +7 ... 12 V für externe Anwendung (max. 200 mA) entnommen, als auch eine externe Spannung von +8 ... 12 V zur Pufferung (max. 2 A) eingespeist werden können.

2. Betriebssystem

Für das Betriebssystem sind 4 KByte ROM reserviert, es umfaßt folgende Komplexe:
- Initialisierung mit RAM- und ROM-Test
- Multiplex-IN/OUT-Routine für Display und Tastatur
- Monitor
- MACRO-Interpreter

Der Anwender kann NMI, Interrupts durch interne SIO und CTC, Interrupts durch externe Systembausteine, Interrupt bei Betätigen einer Taste sowie RST 10H ... RST 28H nutzen, in dem die Anfangsadressen der entsprechenden Routinen in vorgesehene RAM-Zellen eingeschrieben werden.
Bei jedem Monitor-Aufruf und bei jedem Monitor-Neustart durch Betätigen der Taste ESC wird der Monitor-Verursacher angezeigt. Dieser ist vor jedem Aufruf in eine RAM-Zelle zu schreiben. Mögliche Verursacher sind:
- Unterbrechen des Programms durch Betätigen der Taste ESC
- Break
- Operationscode 0FFH
- Error 00 ... 99, frei vom Anwender nutzbar
- DKT-Bus-OUT
- Fehler in einem C-Befehl (siehe MACRO-Befehle)
- Kontrollesefehler in einem L-Befehl (siehe MACRO-Befehle).

Folgende Monitor-Routinen sind implementiert:
- Anzeige des Display-Zustandes vor Monitor-Aufruf
- CPU-Zugriff: Anzeige und Änderung der Register; Rückkehr zum Programm
- Speicherzugriff: Speicherinhalt anzeigen, ändern, fortlaufend schreiben; Starten von Unterprogrammen
- IN/OUT-Zugriff: IN/OUT-Adressen lesen und schreiben
- DKT-Zugriff: Kanäle (siehe MACRO-Befehle) anzeigen und schreiben
- Break: Break anzeigen, löschen, -setzen; Break = PC +/−1 ... 15 setzen und Rückkehr zum Programm
- Speicher füllen mit Konstante oder Blocktransfer
- Freigabe der IFFS-Schnittstelle zum Laden oder Lesen des DKT-Speichers
- Magnetband schreiben oder lesen.

Der Monitor ist hinsichtlich Verursacher und Routinen durch den Anwender erweiterungsfähig.

3. MACRO-Befehle

Die MACRO-Befehle unterstützen die Programmierung in Assemblersprache, sie werden auch vom Betriebssystem genutzt. Der Operationscode jedes MACRO-Befehls besteht aus einem RST-30H-Befehl und nachfolgenden Datenbytes, die den Operationscode und die Operanden enthalten. Zur Assemblierung findet der in /1/ beschriebene MACRO-Assembler Anwendung. Durch Verwendung der Assembleranweisung MACLIST OFF erscheint im Listing neben dem MACRO-Befehl nur der erzeugte Objektcode, so daß diese Befehle vom Anwender wie zusätzliche Maschinenbefehle gehandhabt werden können. Der MACRO-Interpreter benötigt im DKT zur Abarbeitung von n aufeinanderfolgenden MACRO-Befehlen etwa $(n + 1) \times 200\,\mu s$.
Der MACRO-Befehlsvorrat umfaßt vier Basisbefehle:
L – Wertzuweisung
C – Vergleich, bei Ungleichheit Fehlermeldung

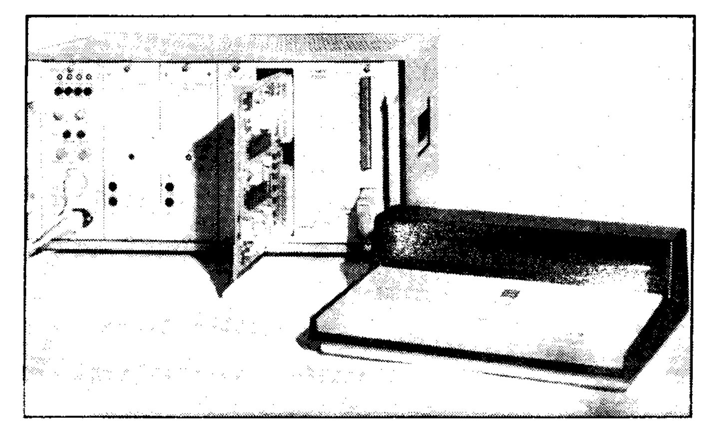


Bild 1 Gesamtansicht des Leiterplattenprüfplatzes
Foto: U. Tschacher

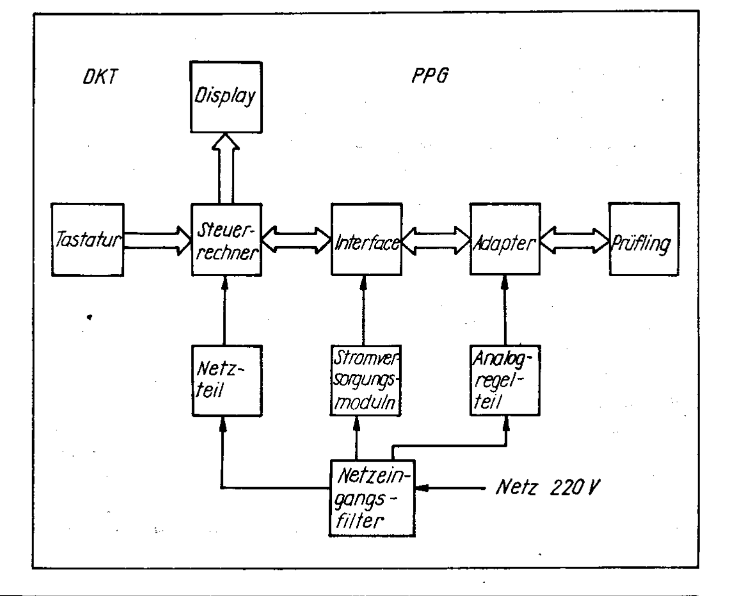


Bild 2 Blockschaltbild

J – Test und bedingter Sprung
W – Wartezeit.

Mittels MACRO-Befehl sind 256 Kanäle adressierbar, sie beinhalten die IN/OUT-Adressen 0 ... 127, die Speicherplätze ([IX + Q] ... [IX + 63]), Display, Tastatur, CPU-Register und ein Steuerregister. Die Kanalwortbreite beträgt 8 Bit.

Markierung und Einzelbitadressierung sowie indirekte Kanaladressierung sind möglich. Jede Displayposition kann sowohl beim Schreiben als auch beim Lesen wahlweise als 7-Segment-Information oder – in Verbindung mit einer Umcodierung Bit 0 ... 3 bzw. Bit 4 ... 7 = Hex-Zeichen – als 7-Segment-Information angesprochen werden.

Programmierbares Prüfgerät

1. Hardware

Die Leiterplatten des programmierbaren Prüfgerätes sind K-1520-kompatibel und werden über eine K-1520-Systembus-Grundverdrahtung betrieben, eine ZRE-Karte ist nicht enthalten. Als Steuerrechner wird vorzugsweise das DKT eingesetzt. Es ist jedoch auch jeder andere Rechner verwendbar, sofern er die Anschlußbedingungen erfüllt. In der hier beschriebenen Minimalvariante sind 8 Datenbit, 4 Adreßbit, ein Schreib-, ein Lese- und ein Rücksetzsignal erforderlich. Zur Stromversorgung der zu prüfenden Leiterplatte sind Regelteile für +5 V und −5,2 V mit einer Belastbarkeit von 30 A eingebaut. Diese Spannungen sind von Hand um ±10% veränderbar. Weiterhin ist eine Leiterplatte mit vier programmierbaren Spannungsquellen enthalten (max. ±18 V; 1 A bzw. ±36 V; 0,5 A). Die Spannungsprogrammierung erfolgt durch den Anwender über Referenzwiderstände, die mit einem Konstantstrom von 1 mA betrieben werden. Eine Überlastung der Spannungsquelle wird durch LEDs signalisiert.

Das Gerät enthält zwei IN/OUT-Leiterplatten mit jeweils 8 Kanälen zu je 8 Bit. Davon sind 7 Kanäle bidirektional, die Aktivierung der OUT-Treiber (DS 8212) erfolgt bei jeder OUT-Operation, die Tristate-Programmierung durch Schreiben auf dem achten Kanal. Dieser steht damit nach außen nur als IN-Kanal zur Verfügung. Die Leitungen eines IN-Kanals sind an einen Schalter mit 8 unabhängig rastenden Tasten geführt. Damit können vom Bearbeiter Meßprogrammparameter eingegeben werden, z. B. die Bestückungsvariante einer Leiterplatte.

Vier Steckplätze sind von außen zugänglich, sie ermöglichen eine Erweiterung des Gerätes und die Aufnahme von Prüfadaptern. Für jeden Prüfling ist ein solcher Adapter erforderlich, der durch Verdrahtung die Zuordnung der vorhandenen IN/OUT-Kanäle und Spannungsquellen zum Prüfling realisiert. Hier ist auch Platz für Hilfsschaltungen (z. B. für Pegelwandler, Lastschaltungen, Impulsformer oder Latches). Insbesondere zur Prüfung von Analogschaltungen sind solche Hardwareunterstützungen notwendig. Für den Adapter stehen Universalleiterplatten mit einem Rasterfeld zur Verfügung.

2. DKT als PPG-Steuergerät

Zur Steuerung des Meßprogrammablaufs sind die Monitorroutinen um folgende Funktionen erweitert:

[illegible]
studierte an der Friedrich-Schiller-Universität Jena Technologie für wissenschaftlichen Gerätebau und arbeitet seit 1976 im Werk Elektronischer Gerätebau des VEB Mikroelektronik „Karl Marx" Erfurt im Prüffeld für digitale Meßgeräte und seit 1983 in der Fertigung und Prüfung automatischer Schaltkreistester.
Ing. Harald Gürth, geboren 1952 in Halle (Saale), erlernte 1969 den Beruf eines Fernmeldemechanikers, erwarb 1980 den Grad eines Ingenieurs für industrielle Elektronik. Er arbeitete von 1974 bis 1978 im Musterbau elektronischer Geräte, anschließend im Prüffeld und seit 1983 in der Inbetriebnahmegruppe des Testerprüffeldes im Werk Elektronischer Gerätebau des VEB Mikroelektronik „Karl Marx" Erfurt. 1985 beendete er ein postgraduales Studium der Mikroprozessortechnik.
Ing. Armin Zaspel, geboren 1951 in Freiberg, erlernte den Beruf eines Rundfunkmechanikers, qualifizierte sich zum Ingenieur für industrielle Elektronik (1980), absolvierte ein postgraduales Studium der Mikroprozessortechnik. Von 1973 bis 1983 arbeitete er im Prüffeld für digitale Meßgeräte und ist seither in der Prüftechnologie des Bereiches Testerbau im Werk Elektronischer Gerätebau des VEB Mikroelektronik „Karl Marx" Erfurt tätig.

RUN: Programmstart bzw. -fortsetzung bis Fehler oder Programmende. Im Fehlerfall wird das Programm gestoppt, der fehlerhafte Zustand bleibt statisch erhalten, die Testnummer und die Adresse des negativ verlaufenden C-Befehls sowie das Vergleichsergebnis werden im Display angezeigt. Der Bearbeiter hat so die Möglichkeit, mit Hilfe des Programmdruckes und der Schaltbilder von Prüfling und Adapterplatte den Fehler zu lokalisieren. Oft genügen dabei Spannungsmessungen im fehlerhaften Zustand. Zusätzliche Unterstützung bieten die weiteren Betriebsarten und Monitor-Routinen (Break, Kanalzugriff usw.).

SHIFT/RUN: Im Unterschied zu RUN wird bei Programmende automatisch neu gestartet. Durch diesen Endloslauf wird die Diagnose zeitweise auftretender Fehler erleichtert, ein Abbruch erfolgt bei Fehler oder nach Betätigung der Taste ESC oder durch RESET. Weiterhin wird der JNE-Befehl nicht ausgeführt, um ein Blockieren des Programmablaufes in Abgleichschleifen zu verhindern.

GO: Programmstart bzw. -fortsetzung bis Testende, der JNE-Befehl wird nicht ausgeführt. Im Fehlerfall wird ein FALL-Flag gesetzt, das Programm wird nicht gestoppt. Nach Testende wird die Testnummer und je nach Zustand des FALL-Flags PASS bzw. FAIL angezeigt. Vorausgesetzt, das Meßprogramm ist so geschrieben, daß die einzelnen Tests voneinander unabhängig lauffähig sind, kann sich der Bearbeiter mit dieser Betriebsart leicht einen Überblick über den Gesamtzustand des Prüflings verschaffen.

SHIFT/GO: Im Unterschied zu GO wird nach Testende derselbe Test neu aufgerufen. Dieser Endloslauf erleichtert die Fehlersuche mittels Oszillograf, ein Abbruch ist mit ESC oder RESET möglich.

NEXT: Programmstart bzw. -fortsetzung, gestoppt wird bei jedem DKT-Bus-OUT (sowohl Maschinen- als auch MACRO-Befehl), bei Testende und bei Programmende. Damit läßt sich das Verhalten des Prüflings im Schrittbetrieb untersuchen.

TNR: Eine Testnummer von 00 bis 99 kann vorgewählt und das Programm gestartet werden.

NAME: Der Meßprogrammname, bestehend aus max. 16 7-Segment-Zeichen, wird angezeigt.

Durch die zusätzlichen Monitor-Routinen sind folgende weitere Monitor-Verursacher erforderlich:

- END, Programmende
- PASS bzw. FAIL, Testende

Außer bei END wird bei allen Monitor-Verursachern zusätzlich die aktuelle Testnummer eingeblendet.

Anwendungskriterien

– Die Geräteanordnung ist aus der Erfahrung entstanden, daß der weit überwiegende Teil der Fehler mit diesem Verfahren erkennbar ist.

– Der beschriebene Prüfplatz eignet sich zur Vorprüfung von Leiterplatten oder elektronischen Baugruppen.

– Auf eine dynamische Endprüfung der Baugruppen wird aufgrund des geringen Restfehleranteils verzichtet. Noch verbleibende dynamische Fehler werden bei der Inbetriebnahme des Gesamtsystems gesucht.

– Eine leichte und schnelle Fehlerdiagnose ergibt sich aus der zugeschnittenen Monitorunterstützung, insbesondere dem Schrittbetrieb.

Literatur

/1/ Stscherbina, A.: MACRO-Assembler für K 1520. Radio, Ferns., Elektron. Berlin 33 (1984) 4, S. 203–204
/2/ Zaspel, A.: Entwicklung eines Steuergerätes zur Prüfung elektronischer Baugruppen. Abschlußarbeit Postgraduales Studium Mikroprozessortechnik TH Karl-Marx-Stadt 1985

Keine Zukunft für Magnetblasenspeicher?

Der US-amerikanische Konzern Intel hat jetzt als sechstes Halbleiter-Unternehmen die Entwicklungsarbeiten an Magnetblasenspeichern aufgegeben. Blasenspeicher werden zwar den digitalen Chips zugerechnet, doch von ihren Eigenschaften her gehören sie eigentlich in das Gebiet der analogen Elektronik, denn die Ausgangssignale von Blasenchips variieren zu stark. Dabei halten sie nicht die Werte ein, die typischen Digitalsignalen vorgegeben sind. Deshalb ist es auch nicht einfach, Blasenspeicher als Digital-Bauelemente einzusetzen. Muster der Blasenspeicher sollen in den USA bisher von der Firma Magnesy ausgeliefert worden sein. Außer von Magnesy werden Magnetblasenspeicher nur noch von den japanischen Konzernen Hitachi und Fujitsu sowie einer französischen Firma produziert, berichtete die Zeitschrift Elektronik.

Programmierung in C

Teil VI

Dr. Thomas Horn
Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden

Im abschließenden Teil dieser Artikelserie werden einfachere und ein komplexeres Programmbeispiel behandelt, an denen vor allem der strukturierte Programmentwurf in der Programmiersprache C verdeutlicht werden soll. Als komplexeres Beispiel wurde ein Druckprogramm gewählt, das nach vorgegebenen Parametern die Seiteneinteilung und die Seitennumerierung ausführt. Die Seitennumerierung kann dabei wahlweise in arabischen und römischen Zahlen erfolgen. Für eine getrennte Numerierung der Kapitel und Anlagen kann vor der Seitennummer ein beliebiger Text stehen, der durch einen Bindestrich getrennt ist, z. B. A-10. Tabulatoren (HT) werden beim Druck durch Leerzeichen ersetzt, wobei der Tabulatorgrundwert frei wählbar ist. Als letztes Beispiel wird ein Programm zur Verarbeitung von Gleitkommazahlen vorgestellt, das eine beliebige Anzahl von Zahlen nach aufsteigender Folge sortieren kann. Zur Vereinfachung des Tests werden die Zahlen im Dialog eingegeben und nach der Sortierung zur Kontrolle gedruckt.

12. Programmbeispiele

12.1. Kopieren eines Textfiles (1. Beispiel)

Es soll ein Programm zum Kopieren eines Textfiles geschrieben werden. Textfiles sind Files, die beliebigen Text im ASCII-Format enthalten. Ein solches Programm kann also auch benutzt werden, um C-Programme auf beliebige Datenträger zu kopieren, u. a. zum Erstellen einer Druckliste auf einem Drucker. Da in diesem Programm die einzelnen Datensätze unverändert kopiert werden sollen, sind hierfür zweckmäßigerweise die E/A-Funktionen **fgets** und **fputs** zu verwenden. Der prinzipielle Programmaufbau leitet sich aus den Prinzipien der Fileverarbeitung ab:

- Eröffnen der verwendeten Files
- Kopieren des Files
- Schließen der verwendeten Files

und ist mit der Struktogrammtechnik als Folge von Strukturblöcken in Bild 1 grafisch verdeutlicht.

Damit das Programm zum Kopieren beliebiger Files benutzt werden kann, soll vor der Fileeröffnung die Filespezifikation des Eingabe- und des Ausgabefiles vom Standardgerät **stdin** angefordert werden. Dazu wird die **printf**-Funktion für die Ausgabe der Eingabeanforderung und die **gets**-Funktion zum Einlesen der Filespezifikation benutzt.

Bei einer fehlerhaften Fileeröffnung wird angenommen, daß die Filespezifikation falsch ist, und die Abfrage der Filespezifikation soll wiederholt werden. Wird auf eine Eingabeanforderung mit **EOF** (Tastenkombination **CTRL/Z**) geantwortet, so soll das Programm abgebrochen werden.

Bei erfolgreicher Beendigung des Kopierens soll die Ausschrift „Kopieren beendet" ausgegeben werden.

Der vollständige Quelltext des Programms **TEST1.C** ist in Bild 2 dargestellt. Bild 3 zeigt den Dialog während der Abarbeitung des Programms zum Kopieren des Programms **TEST1.C** auf den Drucker **LP:**. Bei der ersten Eingabe wurde aus Versehen ein nicht vorhandenes File **TEST.C** spezifiziert. (Der Bildschirmdialog wurde auf einem Drucker als Hardcopy-Gerät protokolliert.)

```
#include <stdio.h>
static char s1[]="Eingabefilespezifikation: ",
            s2[]="Ausgabefilespezifikation: ";
main()
{
char feld[80], sin[25], sout[25];
FILE *fin, *fout;
do {   /* Eroeffnen des Eingabefiles */
   printf("%s", s1);
   if (gets(sin) == NULL) exit();
   fin = fopen(sin, "r");
   if (fin == NULL)
      printf("Fehler in der Filespezifikation\n");
   }
while (fin == NULL);
do {   /* Eroeffnen des Ausgabefiles */
   printf("%s", s2);
   if (gets(sout) == NULL) exit();
   fout = fopen(sout, "w");
   if (fout == NULL)
      printf("Fehler in der Filespezifikation\n");
   }
while (fout == NULL);
/* Kopieren des Files satzweise */
while(fgets(feld, 80, fin)) fputs(feld, fout);
fclose(fout);   /* Schliessen des Ausgabefiles */
fclose(fin);    /* Schliessen des Eingabefiles */
printf("Kopieren beendet\n");
}
```

Bild 2 Programm **TEST1.C** *zum Kopieren eines Textfiles*

```
>run test1
Eingabefilespezifikation: test.c
Fehler in der Filespezifikation
Eingabefilespezifikation: test1.c
Ausgabefilespezifikation: lp:
Kopieren beendet

>
```

Bild 3 Dialog der Abarbeitung des Programms **TEST1.C**

12.2. Kopieren eines Textfiles (2. Beispiel)

Im Quellprogramm des ersten Beispielprogramms ist zu ersehen, daß im Prinzip zweimal die gleiche Anweisungsfolge für die Fileeröffnungen verwendet wurde. Es erhebt sich sofort die Frage, ob man die Fileeröffnungen nicht in einer Funktion durchführen kann, die dann auch später noch in anderen Beispielen genutzt werden kann.

Zu diesem Zweck soll eine Funktion **vfopen** für die Eröffnung eines Files mit variablem Filenamen programmiert werden. Die Funktion benötigt offensichtlich 3 Parameter:

- Zeichenkette der Eingabeanforderung (*anf*)
- Adresse des Filepointers (*fp*)
- Zugriffsart für die Fileeröffnung (*mode*).

Das Programm der Funktion *vfopen* ist in Bild 4 dargestellt. Es gilt hier als besondere Schwierigkeit zu beachten, daß *fp* ein Zeiger

Eingabe der Filespezifikation und Eröffnen des Eingabefiles
Eingabe der Filespezifikation und Eröffnen des Ausgabefiles
while (kein EOF)
Kopieren eines Satzes
Schließen des Ausgabefiles
Schließen des Eingabefiles

Bild 1 Struktogramm des Programms zum Kopieren eines Textfiles

```
#include <stdio.h>
vfopen(anf,fp,mode)
char *anf, *mode;
FILE **fp;
{
char sf[25];
do {
   printf ("%s", anf);
   if (gets(sf) == NULL) exit();
   if ((*fp = fopen(sf, mode)) != NULL) return;
   printf("Fehler in der Filespezifikation\n");
   }
while(1);
}
```

Bild 4 Programm der Funktion **vfopen**

```
#include <stdio.h>
main()
{
char feld[80];
FILE *fin, *fout;
/* Eroeffnen der Files */
vfopen("Eingabefilespezifikation: ",&fin,"r");
vfopen("Ausgabefilespezifikation: ",&fout,"w");
/* Kopieren des Files satzweise */
while(fgets(feld, 80, fin)) fputs(feld, fout);
fclose(fout);   /* Schliessen des Ausgabefiles */
fclose(fin);    /* Schliessen des Eingabefiles */
printf("Kopieren beendet\n");
}
```

Bild 5 Programm **TEST2.C** *zum Kopieren eines Textfiles*

auf einen Filepointer ist. Deshalb ist er auch in Zeile 4 als Zeiger auf einen Zeigertyp definiert, und in Zeile 10 wird über den Zeiger *fp* auf den Filepointer indirekt zugegriffen. Durch Anwendung der Funktion **vfopen** ergibt sich eine wesentliche Vereinfachung des Programms **TEST1.C**, das als Programm TEST 2.C in Bild 5 dargestellt ist. Dieses Beispiel macht deutlich, wie sich der modulare Aufbau von größeren Programmsystemen auf ihre Transparenz, Einfachheit und Fehlerfreiheit auswirkt.

12.3. Kopieren eines Textfiles (3. Beispiel)

In diesem Beispiel soll die gleiche Aufgabenstellung noch einmal gelöst werden. Im Unterschied zu den ersten beiden Beispielen soll jetzt aber der Dialog zur Abfrage der Filespezifikationen durch die Übernahme der Filespezifikationen aus der Kommandozeile ersetzt werden.
Das Programm soll wie folgt gestartet werden:

>**TEST3** *infile outfile*

Wenn die Ausgabefilespezifikation *outfile* nicht angegeben wird, soll die Ausgabe auf das Standardausgabegerät **stdout** erfolgen. Wird kein Parameter spezifiziert, so soll das Programm ohne ein Resultat beendet werden. Für das Beispiel aus 12.1. zum Kopieren des Files **TEST1.C** auf den Drucker **LP:** müßte dann die Kommandozeile wie folgt spezifiziert werden:

>TEST3 TEST1.C LP:

Der vollständige Programmtext ist in Bild 6 dargestellt und zeigt, wie die Parameter aus der Kommandozeile zu verwenden sind.

12.4. Drucken eines Textfiles mit Seitennumerierung

Aufbauend auf den ersten einfachen Beispielen soll nun das Kopieren des Textfiles um zwei Funktionen erweitert werden:

- Im Text enthaltene Tabulatoren sollen durch Leerzeichen ersetzt werden. Diese Funktion ist nützlich, da erstens viele Drucker das Steuerzeichen „Tabulatorsprung" (*HT* – **011**) nicht interpretieren können und zweitens man die Möglichkeit zum Steuern des Tabulatorsprungs hat, um wieviele Druckpositionen maximal gesprungen werden soll.

```
#include <stdio.h>
main(argc, argv)
int argc; char *argv[];
{
char feld[80];
FILE *fin, *fout;
/* Eroeffnen der Files */
if (argc == 1) exit();
if ((fin = fopen(argv[1], "r")) == NULL)
   error ("Eingabefile kann nicht eroeffnet werden\n");
if (argc == 2) fout = stdout;
else if ((fout = fopen(argv[2], "w")) == NULL)
   error ("Ausgabefile kann nicht eroeffnet werden\n");
/* Kopieren des Files satzweise */
while(fgets(feld, 80, fin)) fputs(feld, fout);
fclose(fout);  /* Schliessen des Ausgabefiles */
fclose(fin);   /* Schliessen des Eingabefiles */
printf("Kopieren beendet\n");
}
```

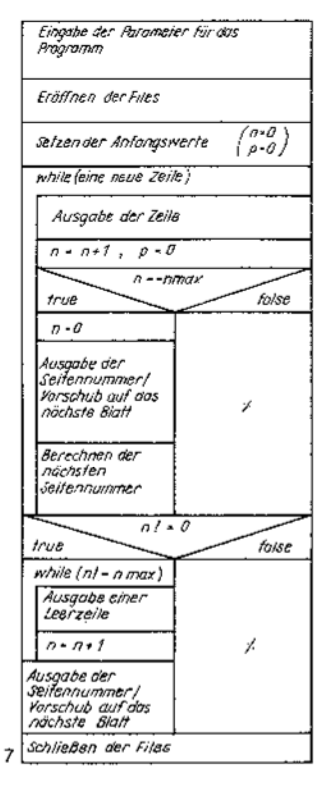
Bild 6 Programm **TEST3.C** *zum Kopieren eines Textfiles*

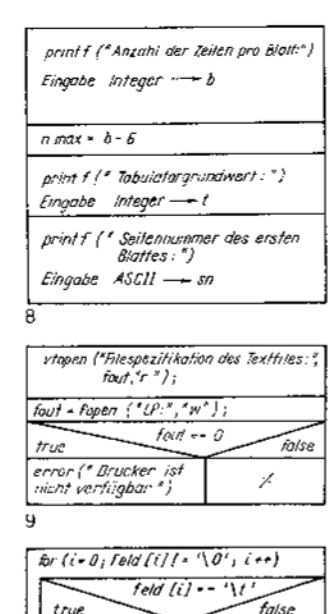
Bild 7 Struktogramm des Programms zum Drucken von Textfiles mit Seitennumerierung

Bild 8 Struktogramm für die Eingabe der Programmparameter

Bild 9 Struktogramm für das Eröffnen der Files

Bild 10 Struktogramm für die Ausgabe einer Zeile





Bei den Rechenanlagen des SKR ist dieser Tabulatorgrundwert konstant 8, was für eine Reihe von Anwendungen nicht immer zweckmäßig ist.

- Ausgehend vom eingestellten Zeilenabstand (4 Zeilen/Zoll, 6 Zeilen/Zoll usw.) und von der Papiergröße soll beim Ausgeben des Textfiles eine Seiteneinteilung erfolgen und auf jeder Seite unten in der Mitte eine Seitennummern gedruckt werden.

Da größere Texte zweckmäßigerweise in mehrere Files unterteilt werden, muß die Seitennummer der ersten Seite frei wählbar sein. Die Seitennummer der Folgeseiten sollen automatisch gebildet werden.
Gleichzeitig sollen neben arabischen auch römische Seitennummern sowie eine getrennte Numerierung nach Abschnitten, Kapiteln bzw. Anlagen zugelassen werden, indem vor der variablen Seitennummer durch Bindestrich getrennt ein konstanter Text stehen darf.
In der ersten Entwurfsetappe werden die Funktionen des Programms im groben spezifiziert, wie Eingabe der Parameter des Programms, Eröffnen der Files, Setzen der Anfangswerte für den Zeilenzähler **n** und den Druckpositionszähler **p**, Kopieren des Textfiles unter Beachtung der zwei geforderten Funktionen, Auffüllen der letzten Seite mit Leerzeilen und Schließen der Files.
Charakteristisch für das Programm sind vor allem der vierte und fünfte Strukturblock. Der vierte Strukturblock besteht aus einer Abweisschleife, die solange ausgeführt wird, wie noch eine Textzeile verfügbar ist. Unter Beachtung der Substition der Tabulatoren ist die Textzeile auszugeben. Nach der Ausgabe ist der Zeilenzähler **n** zu inkrementieren und der Positionszähler **p zu löschen. Wenn die maximale Zeilenzahl erreicht wurde, muß** der Zeilenzähler **n** gelöscht werden, die Seitennummer gedruckt werden, ein Vorschub auf das nächste Blatt erfolgen und die nächste Seitennummer berechnet werden.
Im fünften Strukturblock wird der Zeilenzähler **n** getestet. Bei **n** ungleich **0** ist die letzte Seite unvollständig. Solange, bis **n** gleich **nmax** ist, werden Leerzeilen ausgegeben. Anschließend wird die Seitennummer gedruckt und ein Vorschub auf das nächste Blatt realisiert. Das Strukturprogramm für diesen grundsätzlichen Programmablauf ist in Bild 7 dargestellt.
Im weiteren werden wir nun dieses Struktogramm schrittweise verfeinern. Die Eingabe der Programmparameter (Bild 8) wird durch die einzugebenden Parameter konkretisiert:

– Anzahl der Zeilen pro Blatt **b**. Für die Seitennummer, den unteren und den oberen Blattrand sollen **6** Zeilen verwendet werden, so daß sich **nmax** aus **b–6** ergibt.
– Tabulatorgrundwert **t**.
– Seitennummer des ersten Blattes **sn**.

Das Eröffnen der Files (Bild 9) wird durch die zu eröffnenden Files konkretisiert. Für das Eröffnen des Textfiles wird die Funktion **fopen()** verwendet. Die Druckereröffnung erfolgt ganz normal über die Funktion **vfopen()**. Die Ausgabe der Zeile (Bild 10) erfolgt zeichenweise. Jedes Zeichen wird auf *HT*(**\t**) analysiert. Bei einem *HT* wird ein Leerzeichen ausgegeben. Ist die neue Position nicht durch den Tabulatorgrundwert teilbar, so werden weitere Leerzeichen ausgegeben.
Analog können alle anderen Strukturblöcke konkretisiert und formalisiert werden. Im Ergebnis dieser zweiten Programmentwurfsetappe entsteht dann das in Bild 11 dargestellte verfeinerte Struktogramm. Die Ausgabe der Seitennummer und des Blattwechsels kann mit einer **fprintf**-Anweisung erfolgen, wenn eine entsprechende Textkonstante **tc** mit den erforderlichen Steuerzeichen benutzt wird. Die Berechnung der

neuen Seitennummer ist ein recht komplexes und im Prinzip auch selbständiges Problem, so daß festgelegt wurde, eine Funktion **seite()** dafür zu definieren.
Das dem Bild 11 äquivalente C-Programm ist in Bild 12 dargestellt.
Die Funktion **seite()** dient der Berechnung einer neuen Seitennummer (Bild 13). Bei einer leeren Zeichenkette ist keine Erhöhung der Seitennummer erforderlich, und es erfolgt sofort ein Rücksprung. Nach diesem Test muß ein Bindestrich gesucht werden. Wurde ein Bindestrich gefunden, so wird der Index **i** auf das nächste Zeichen nach dem Bindestrich gesetzt. Anderenfalls wird der Index **i** auf das nächste Zeichen nach dem Bindestrich gesetzt. Ist das erste Zeichen, worauf der Index **i** zeigt, eine Ziffer von **'0'–'9'**, so liegt eine arabische Zahl vor, anderenfalls eine römische Zahl. Die arabische Zahl wird gleich in der ASCII-Zeichenkette erhöht. Es wird die letzte Ziffer gesucht und inkrementiert. Wenn der Kode größer wird als der Kode der Ziffer **'9'**, so wird die Ziffer auf **'0'** gesetzt und die vorhergehende Ziffer inkrementiert usw. Wenn die Zahl um eine Ziffer größer wird, so muß sie um ein Zeichen nach rechts verschoben werden, und an vorderster Stelle muß eine **'1'** eingeblendet werden.

Das Inkrementieren einer römischen Zahl ist wesentlich komplizierter. Deshalb wurde hier der Weg der Konvertierung einer römischen Zahl in einen Binärwert (Integer), des Inkrementierens des Binärwertes und der Konvertierung des Binärwertes in die römische Zahlendarstellung gewählt. Die Konvertierung aus der römischen bzw. in die römische Zahlendarstellung sind universell anwendbare Algorithmen. Darum wurden diese Algorithmen als gesonderte Funktionen implementiert:

crtb() – Konvertierung aus dem Römischen ins Binäre
cbtr() – Konvertierung aus dem Binären ins Römische

Die Funktionen **seite()**, **crtb()** und **cbtr()** sind in den Bildern 14, 15 und 16 dargestellt.
Bild 17 zeigt den Dialog der Abarbeitung des Programms **druck**. Als Beispiel wurde ein Programm **sort.c** ausgedruckt. Das Bild 18 zeigt die Ausgabe des Programms **sort.c** mit dem Tabulatorgrundwert **2** und Bild 19 mit dem Tabulatorgrundwert **5**. Die Anweisungen in der **for**-Schleife wurden jeweils mit 1, 2 oder 3 Tabulatorsprüngen (*HT*) eingerückt. Die Seitennumerierung ist in Bild 12 gezeigt.

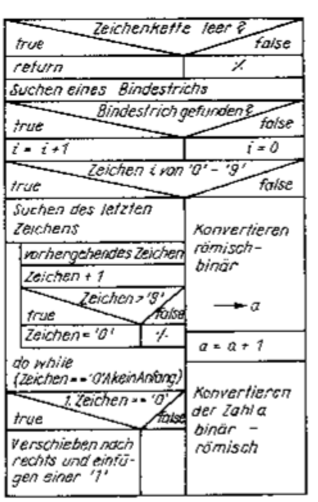


Bild 13 Struktogramm des Moduls für die Erhöhung der Seitennummern

12.5. Sortieren von Zahlen

Zum Abschluß sei ein kleines Programm zur Verarbeitung von **float**-Zahlen am Beispiel eines häufig angewendeten Algorithmus zum

```
printf ("Anzahl der Zeilen pro Blatt :")
Eingabe Integer ──► b
nmax = b - 6
printf (" Tabulatorgrundwert : ")
Eingabe Integer ──► t
printf (" Seitennummer des ersten Blattes : ")
Eingabe ASCII ──► sn
vfopen ("Filespezifikation des Textfiles",fout,"w"
fout = fopen (" LP: ", "w")
fout == 0 : true | false
  true:  error (" Drucker ist nicht verfügbar")
  false: ./.
n = 0 , P = 0
while (fgets (feld , 100 , fin))
  for (i = 0 ; feld [i] ! = '\0' ; i++
    feld [i] = = 't' : true | false
      true:  putc ('b', fout)
             p = p + 1
             while (p % t ! = 0
               putc ('b', fout)
               p = p + 1
      false: putc (feld[i], fout)
             p = p + 1
  n = n + 1 , p = 0
  n = = nmax : true | false
    true:  n = 0
           fprintf (fout , tc , sn)
           seite (sn)
    false: ./.
n ! = 0 : true | false
  true:  while (n ! = nmax)
           putc ('\n', fout)
           n = n + 1
         fprintf (fout , tc , sn)
  false: ./.
fclose (fin) , fclose (fout)
```

Bild 11 Verfeinertes Struktogramm des Programms zum Drucken von Textfiles

```
static char tc[]="\n\n                \
                  %s\n\n\n\n";
#include <stdio.h>
main()
/* Programm zum Drucken von Textfiles
mit Seitennumerierung */
{
/* Vereinbarung der Pointer, Variablen, Felder */
FILE *fin,*fout;
int n,nmax,b,t,p,i;
char sn[10],feld[100],conv[20];
/* Eingabe der Programmparameter */
printf("Anzahl der Zeilen pro Blatt: ");
gets(conv);
sscanf(conv,"%d",&b);
nmax=b-6; /* druckbare Zeilen pro Blatt */
printf("Tabulatorgrundwert: ");
gets(conv);
sscanf(conv,"%d",&t);
printf("Seitennummer des ersten Blattes: ");
gets(sn);
/* Eroeffnen des Textfiles */
vfopen("Filespezifikation des Textfiles: ",&fin,"r");
/* Eroeffnen des Druckers */
if((fout=fopen("LP:","w"))==NULL)
     error("Drucker ist nicht verfuegbar");
/* Setzen der Anfangswerte */
n=0;  /* Zeilenzaehler */
p=0;  /* Positionszaehler */
/* Ausgabe des Textes mit Aufbereitung */
while(fgets(feld,100,fin))  /* Zeile einlesen */
{
     /* Ausgabe einer Zeile */
     for(i=0;feld[i]!='\0';i++)
     /* Analyse auf Tabulator */

                        6–1

          if(feld[i]=='\t')
          {
               /* Ausgabe eines Leerzeichens und
               positionieren auf die naechste durch
               t teilbare Position */
               putc(' ',fout); p++;
               while(p%t!=0) { putc(' ',fout);p++; }
          }
          else
          {
               /* Ausgabe aller anderen Zeichen */
               putc(feld[i],fout);p++;
          }
     n++;        /* Zeilenzaehler +1 */
     p=0;        /* Positionszaehler =0 */
     /* Analyse auf Blattende */
     if(n==nmax)
     {
          n=0;  /*Zeilenzaehler loeschen */
          fprintf(fout,tc,sn); /*Drucken Seitennummer*/
          seite(sn);  /* Erhoehen der Seitennummer */
     }
}
/* Auffuellen der letzten Seite mit Leerzeilen */
if(n!=0)
{
     while(n!=nmax)
          {  putc('\n',fout);n++; }
     /* Drucken der letzten Seitennummer */
     fprintf(fout,tc,sn);
}
/* Schliessen der Files */
fclose(fin); fclose(fout);
}

                        6–2
```

Bild 12 Text des Programms zum Drucken von Textfiles

```
seite(p)
/* Berechnung der neuen Seitennummer */
/* p -- Textpuffer mit Seitennummer */
char p[];
{
int i, j, a;
if (p[0]=='\0') return; /*keine Seitennummer*/
for (i = 0; p[i] != '\0' && p[i] != '-'; i++);
/* wenn Seitennummer mit "-", dann Index i
auf naechstes Zeichen nach "-" */
if (p[i] == '-') i++; else i = 0;
/* Test auf arabisch, ansonsten roemisch */
if ((p[i] >= '0') && (p[i] <= '9'))
{ /* arabisch */
     /* Suchen des Zeichenkettenendes */
     for (j=i; p[j] != '\0'; j++);
     /* Erhoehen der letzten Ziffer, wenn =0,
     dann erhoehen der vorletzten Ziffer usw.*/
     do
     {
          j--; p[j]++;
          if(p[j]> '9') p[j]= '0';
     }
     while (( p[j] == '0') && ( j > i));
     /* wenn erste Ziffer =0, dann verschieben
     nach rechts und einschieben einer 1 */
     if ((j == i) && (p[j] == '0'))
     {
          for (j = i; p[j] != '\0'; j++);
          for (;j >= i; j--) p[j+1]= p[j];
          p[i] = '1';
     }
}
else
{ /* roemisch */
     a=crtb(p,i);
     a++;
     cbtr(a,p,i);
}
return;
}
```

Bild 14 Text des Moduls für die Erhöhung der Seitennummern

```
crtb(p,i)
/* Konvertierung Roemisch in Binaer */
/* p -- Textpuffer mit roemischer Zahl
   i -- Anfangsposition im Textpuffer
   Funktionswert -- aequivalenter Binaerwert */
int i; char p[];
{
int a;
a=0;
/* Tausender */
while (p[i]=='M') { i++; a+=1000; }
/* Hunderter */
if ((p[i]=='C') && (p[i+1]=='M')) { i+=2; a+=900; }
if ((p[i]=='C') && (p[i+1]=='D')) { i+=2; a+=400; }
if (p[i]=='D') { i++; a+=500; }
while (p[i]=='C') { i++; a+=100; }
/* Zehner */
if ((p[i]=='X') && (p[i+1]=='C')) { i+=2; a+=90; }
if ((p[i]=='X') && (p[i+1]=='L')) { i+=2; a+=40; }
if (p[i]=='L') { i++; a+=50; }
while (p[i]=='X') {i++; a+=10; }
/* Einer */
if ((p[i]=='I') && (p[i+1]=='X')) { i+=2; a+=9; }
if ((p[i]=='I') && (p[i+1]=='V')) { i+=2; a+=4; }
if (p[i]=='V') { i++; a+=5; }
while (p[i]=='I') { i++; a+=1; }
return (a);
}
```

Bild 15 Text des Programms für die Konvertierung Römisch–Binär

```
cbtr(a,p,i)
/* Konvertierung Binaer in Roemisch */
/* a -- zu konvertierender Wert
   p -- Textpuffer fuer die roemische Zahl
   i -- Anfangsposition im Textpuffer */
int a, i; char p[];
{
/* Tausender */
while (a>=1ØØØ) { a-=1ØØØ; p[i++]='M'; }
/* Hunderter */
if (a>=9ØØ) { a-=9ØØ; p[i++]='C'; p[i++]='M'; }
if (a>=5ØØ) { a-=5ØØ; p[i++]='D'; }
if (a>=4ØØ) { a-=4ØØ; p[i++]='C'; p[i++]='D'; }
while (a>=1ØØ) { a-=1ØØ; p[i++]='C'; }
/* Zehner */
if ( a>=9Ø) { a-=9Ø; p[i++]='X'; p[i++]='C'; }
if (a>=5Ø) { a-=5Ø; p[i++]='L'; }
if (a>=4Ø) { a-=4Ø; p[i++]='X'; p[i++]='L'; }
while (a>=1Ø) { a-=1Ø; p[i++]='X'; }
/* Einer */
if (a>=9) { a-=9; p[i++]='I'; p[i++]='X'; }
if (a>=5) { a-=5; p[i++]='V'; }
if (a>=4) { a-=4; p[i++]='I'; p[i++]='V'; }
while (a>=1) { a-=1; p[i++]='I'; }
p[i++]='\Ø';  /* Zeichenkettenende */
return;
}
```

Bild 16 Text des Programms für die Konvertierung Binär–Römisch

```
>run druck
Anzahl der Zeilen pro Blatt: 68
Tabulatorgrundwert: 5
Seitennummer des ersten Blattes: 1
Filespezifikation des Textfiles: sort.c
>
```

Bild 17 Eingabedialog des Programms zum Drucken von Textfiles

```
main()
/* Sortieren von max 100 Zahlen */
{
int i,j,k;
static float a[100], z;
printf ("Anzahl der Zahlen: ");
scanf ("%d", &k);
printf ("Geben sie %d Zahlen getrennt durch\
 SP, HT und/oder CR ein:\n", k);
for (i=0; i<k; i++) scanf("%f", &a[i]);
for (i=0; i<k-1; i++)
  for (j=i+1; j<k; j++)
    if (a[i]>a[j])
      {
      z=a[i];
      a[i]=a[j];
      a[j]=z;
      }
printf ("Ergebnisse:\n");
for (i=0; i<k; i++) printf ("  %f\n",a[i]);
}
```

Bild 18 Programm sort.c mit Tabulatorgrundwert 2 gedruckt

```
main()
/* Sortieren von max 100 Zahlen */
{
int i,j,k;
static float a[100], z;
printf ("Anzahl der Zahlen: ");
scanf ("%d", &k);
printf ("Geben sie %d Zahlen getrennt durch\
 SP, HT und/oder CR ein:\n", k);
for (i=0; i<k; i++) scanf("%f", &a[i]);
for (i=0; i<k-1; i++)
     for (j=i+1; j<k; j++)
          if (a[i]>a[j])
               {
               z=a[i];
               a[i]=a[j];
               a[j]=z;
               }
printf ("Ergebnisse:\n");
for (i=0; i<k; i++) printf ("  %f\n",a[i]);
}
```

Bild 19 Programm sort.c mit Tabulatorgrundwert 5 gedruckt

Sortieren von Zahlen gezeigt. Zum Testen des Algorithmus können die Tabellengröße und die Testzahlen mit **scanf**-Anweisungen eingelesen werden. Nach der Sortierung wird die Tabelle mit **printf**-Anweisungen ausgegeben.

Der Text des Sortierprogramms ist in Bild 18 und 19 abgedruckt. Von der Vergleichsoperation in Zeile 13 ist abhängig, ob die Zahlen in steigender oder fallender Folge sortiert werden sollen („größer als" – steigende Folge).

13. Schlußbemerkungen

In der vorliegenden Artikelreihe „Programmierung in C" wurde der Versuch unternommen, dem Leser in einer methodisch-didaktisch aufbereiteten Form die international weit verbreitete Programmiersprache C vorzustellen. Die Artikelreihe kann dem Leser deshalb als Selbststudienmaterial dienen, wobei gleichzeitig auch auf Vollständigkeit Wert gelegt wurde, so daß es auch für die praktische Arbeit genutzt werden kann. Die abschließenden Beispiele verfolgten das Ziel, dem Leser bei der Programmierung eigener Aufgaben eine methodische Unterstützung zu geben.

Abschließend möchte ich hoffen, mit dieser Artikelreihe den Interessen vieler Leser entgegengekommen zu sein, eine Anregung zur intensiveren Beschäftigung mit der Programmiersprache C gegeben und somit auch einen Beitrag zu ihrer weiteren Verbreitung in unserer Republik geleistet zu haben.

✉ KONTAKT ☏

Informatikzentrum des Hochschulwesens
an der Technischen Universität Dresden,
Mommsenstr. 13, Dresden, 8027; Tel. 4575303.

Der Autor unserer sechsteiligen Beitragsfolge zu C, Doz. Dr. sc. techn. Thomas Horn (39), studierte von 1966–1972 an der Elektrotechnischen Hochschule „W. I. Uljanow (Lenin)" in Leningrad in der Fachrichtung Elektronische Rechenanlagen. Von 1972 bis 1975 promovierte er an der gleichen Hochschule auf dem Gebiet der Betriebssysteme zum Dr.-Ing.
Seit 1975 ist er an der Ingenieurhochschule Dresden auf dem Gebiet der Software für Klein- und Mikrorechner tätig. 1983 promovierte er an der Technischen Universität Dresden auf dem Gebiet der Architektur von Betriebssystemen für Klein- und Mikrorechner zum Dr. sc. techn. 1985 wurde er an der Ingenieurhochschule Dresden zum Hochschuldozenten für die Programmierung von Mikrorechnern berufen. Seit der Gründung des Informatikzentrums des Hochschulwesens ist Doz. Dr. Horn an der Technischen Universität Dresden beschäftigt.
Sein Arbeitsgebiet liegt gegenwärtig vorrangig auf dem Gebiet der Gestaltung und Implementierung von Systemsoftware für moderne Kleinrechner. Dabei kommt der Anwendung der Systemprogrammiersprache C eine besondere Bedeutung zu.
Doz. Dr. Horn ist seit 1981 Leiter der Arbeitsgruppe Software und seit 1983 Vorsitzender des Problemrates der Kooperationsgemeinschaft SM3/SM4.

C-Programmierwettbewerb in MP

Ihr erworbenes Wissen zur Programmiersprache C können Sie in unserem Programmierwettbewerb, zu dem wir in MP 7/87, 2. Umschlagseite aufrufen, unter Beweis stellen. Zwei Aufgaben, die zu lösen sind, hat unser Autor für Sie ausgesucht. Unter den richtigen Einsendungen verlosen wir 10 Bücher des Titels „UNIX und C", der in diesem Jahr in unserem Verlag erschienen ist. *MP*

Die Programmiersprache C

Folienreihen HFR 913/914 als Lehrmittel für die Aus- und Weiterbilung

Vom Institut für Film, Bild und Ton werden ab 1988 zwei Folienreihen für die Aus- und Weiterbildung zur „Programmierung in C" mit einem Gesamtumfang von 30 Polyluxfolien angeboten. Die Folienreihen sind so gestaltet, daß sie sowohl im Hochschulstudium als auch zur Weiterbildung in Betriebsakademien und KDT-Lehrgängen für Vorlesungen und Übungen einsetzbar sind. Insbesondere dienen sie zur methodischen Stützung einer kompletten Vorlesungsreihe zur Programmierung in C.
Spezifika einzelner C-Compiler unter bestimmten Betriebssystemen spiegeln sich in den Folienreihen nicht wider.
Die Folienreihe HFR 913 behandelt die Grundsprachelemente, die einfachen Datentypen, Operatoren, Ausdrücke und Steueranweisungen.
Die Folienreihe HFR 914 erläutert die höheren Datentypen wie Zeiger, Felder, Strukturen und Vereinigungen sowie die Funktionen im allgemeinen, die Ein- und Ausgabefunktionen im speziellen und den Makropräprozessor.
Die Folienreihen können ab sofort beim
VEB DLK Erfurt
Audiovisuelle Lehrmittel
Junker-Jörg-Str. 31
Berlin
1162
bestellt werden.

Was erscheint demnächst in MP-Kurs?

In Heft 7/87 beginnen wir mit einer Beitragsreihe über *Programmierung in MACRO-SM*. Bereits in MP 8/87 wird ebenfalls mit einer neuen Folge, nämlich über REDABAS, begonnen. Eine Artikelserie wird also in der Regel nicht fortlaufend gedruckt, sondern alternierend mit anderen. Wir hoffen, damit mehr den Interessen aller Leser zu entsprechen. So muß ein Leser u. U. nicht mehr Monate „warten", bis ein für ihn interessantes Thema unter dieser Rubrik erscheint. Weiterhin werden u. a. Artikelfolgen vorbereitet zum Mikroprozessor(-system) K 1810 WM 86 (8086) und zu Turbo Pascal. *MP*

Nachladbare Gerätetreiber für Personal- und Bürocomputer

Joachim Geiler
Ingenieurhochschule Mittweida, ORZ

In CP/M-kompatiblen Betriebssystemen (zum Beispiel CP/A, DAC, SCP) ist eine Änderung bzw. Ergänzung von Gerätetreibern normalerweise nur durch Änderung des BIOS möglich. Im folgenden Beitrag wird beschrieben, wie man derartige Treiber nachladbar gestalten kann und ein Programmsystem zur Anzeige, Aktivierung und Deaktivierung derartiger Treiber vorgestellt.

1. Wichtige Adressen

CP/M-kompatible Betriebssysteme bestehen aus den drei Teilen CCP (Console Command Processor), BDOS (Basic Disk Operating System) und BIOS (Basic Input/Output System). Im BIOS befinden sich die standardmäßig vorhandenen Treiberroutinen für Tastatur, Drucker, Bildschirm und Diskettenlaufwerke. Die Lage von BDOS und BIOS wird Anwenderprogrammen durch zwei Sprungbefehle bekannt:
Auf der Adresse 0 steht ein Sprung zum Warmstartsprung der BIOS-Sprungleiste. Auf Adresse 5 steht ein Sprung zum ersten Befehl des BDOS. Die meisten Programme benutzen diese Angabe, um festzustellen, wieviel Platz (TPA) ihnen zur Verfügung steht. Sie überlagern dann den CCP und führen am Ende einen Warmstart aus, um den CCP nachzuladen. Genau 806H vor dem ersten BDOS-Befehl befindet sich der Einsprung in den CCP. Am Anfang des BDOS stehen hinter einem Sprungbefehl die vier Adressen für die Fehlerbehandlungsroutinen bei den Fehlern Bad Sector, Select, R/O und File R/O. Diese Adressen werden von Programmen mit eigener Fehlerbehandlung (z. B. POWER) modifiziert.

2. Anforderungen an einen nachladbaren Gerätetreiber

Ein nachladbarer Gerätetreiber sollte folgenden Forderungen gerecht werden:

- Unabhängigkeit vom speziellen Betriebssystem, daraus folgt eine freie Verschieblichkeit auf beliebige Adressen im RAM
- Speicherresidenz auch nach einem Warmstart
- Möglichkeit des Deaktivierens (d. h. Verwendung des entsprechenden BIOS-Treibers) und des Aktivierens des Teibers
- Möglichkeit des Streichens des Treibers, ohne daß ein Kaltstart notwendig ist
- Möglichkeit des Ladens mehrerer solcher Treiber in beliebiger Reihenfolge
- volle Funktionstüchtigkeit aller Betriebssystemfunktionen bei geladenen Treibern.
- Möglichkeit der Anzeige aller geladenen Treiber.

Der im folgenden beschriebene Treiberrahmen wird allen diesen Forderungen gerecht und ist unter allen Betriebssystemen, deren Aufbau Bild 1 entspricht und unter SCP 1715 lauffähig.

3. Realisierung

3.1. Verschieblichkeit

Um eine beliebige Verschieblichkeit des Treibercodes zu realisieren, wird der eigentliche Treiber mit Hilfe der .PHASE-Klausel des Assemblers jeweils einmal auf Adresse 0 (ergibt drv0.rel) und einmal auf Adresse 201H (ergibt drv201.rel) übersetzt. Ein spezieller Lader kann so durch Vergleich beider Codes die bei der Verschiebung zu verändernden Bytes feststellen und den Treiber auf jede beliebige Adresse laden. Absolute Adressen sind bei beiden Codes gleich und werden deshalb beim Laden nicht verändert. Lader, drv0 und drv201 werden in dieser Reihenfolge gelinkt.

3.2. Treiberaufbau

Ein nachladbarer Treiber wird unmittelbar unter den CCP geladen. Ähnlich den Debuggern DU und ZSID schützt sich auch der Treiber vor Überschreiben durch ein Anwenderprogramm, indem er den Sprungbefehl auf Adresse 5 verändert. Der erste Befehl des Treibers wird durch den Sprungbefehl von Adresse 5 ersetzt und dieser durch einen Sprung zum Treiber modifiziert. Bei mehreren geladenen Treibern sind bei einem BDOS-Aufruf also zunächst Sprünge von einem Treiber zum anderen auszuführen.
Nach diesem Sprungbefehl müssen die vier Adressen der Fehlerbehandlungsroutinen stehen, damit auch Programme wie POWER einwandfrei laufen. Die Fehleradressen werden vom Lader mit den Fehleradressen des BDOS (bzw. des vorherigen Treibers) getauscht. Die Fehlerbehandlungsroutinen des Treibers bestehen aus Sprüngen zu den entsprechenden Fehleradressen. Somit erfolgen bei BDOS-Fehlern wiederum zunächst Sprünge von einem Treiber zum anderen und dann in die Fehlerbehandlung des BDOS bzw. Nutzerprogrammes.
Jeder Treiber beinhaltet eine (fast) komplette BIOS-Sprungleiste mit den Sprüngen JP WBOOT bis JP SECTRAN /1/. Alle BIOS-Ansprünge, die vom Treiber nicht verändert werden sollen, stehen als „Sprünge auf sich selbst" in der Treibersprungleiste. Vom Lader wird nun nach erfolgtem Verschieben (und damit Verändern der Adreßteile der Treibersprungleiste) die Treibersprungleiste mit der BIOS-Sprungleiste vertauscht. Damit wird nach Aufruf der BIOS-Funktion, die durch den Zusatztreiber realisiert werden soll, sofort zu dieser Funktion verzweigt. Im anderen Fall wird in Sprüngen, die auf allen geladenen Treibern aufsetzen, ins BIOS gesprungen.
Im Treiber sind mit fester Distanz zum Treiberanfang 2 Byte als Statusbits und 15 Byte für einen Treiberidentifikator vorgesehen, der z. B. durch das Hilfsprogramm DRVUTL angezeigt werden kann. Die Bits 0 bis 1 des ersten Statusbyte sind fest vergeben. Bit 0 ist 1, wenn der Treiber deaktiviert ist, das heißt, wenn auf den entsprechenden Treiber des Betriebssystems zurückgeschaltet werden kann. Das ist immer dann der Fall, wenn vom Treiber keine E/A-Tore des Betriebssystems neu initialisiert werden. Das Bit 1 des Statusbyte ist 1, wenn der Treiber aktiv ist.
Ein alternativer Druckertreiber beginnt dann z. B. mit folgender Befehlsfolge:

```
;in Treibersprungleiste
  .
  .
conout: jp conout
list: jp nlist
  .
  .
; im Treiber:
nlist:  ld hl, status
        bit 1, (hl)    ;Treiber aktiv?
        jp z, list     ;wenn nein, dann
                       ;list-Treiber des
                       ;Betriebssystems
                       ;benutzen!
```

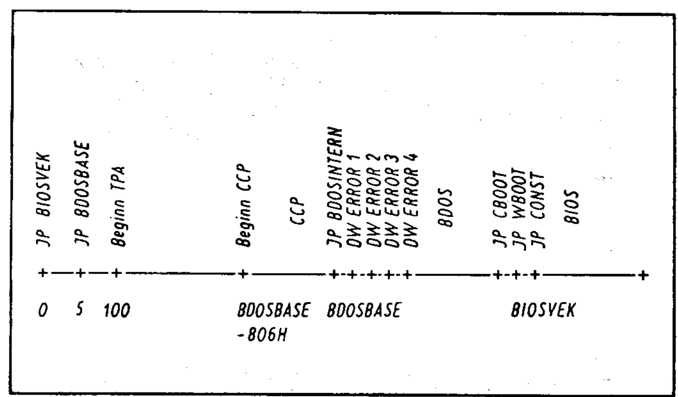


Bild 1 Speicheraufteilung unter CP/M

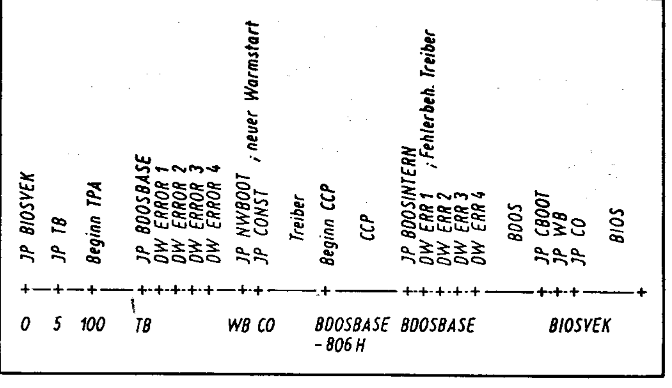


Bild 2 Speicheraufteilung bei geladenem Treiber

Joachim Geiler (37) studierte von 1968 bis 1972 an der TU Dresden Informationstechnik. Seit 1972 an der IH Mittweida beschäftigt, ist er von 1979 an Mitarbeiter am dortigen Rechenzentrum. Seit 1981 befaßt er sich schwerpunktmäßig mit der Softwareentwicklung für Industrierobotersteuerungen. Sein besonderes Interesse gilt daneben der Systemsoftware für Büro- und Personalcomputer.

Jeder Treiber besitzt eine eigene Warmstartroutine, da von der Warmstartroutine des BIOS die BDOS-Sprungadresse wiederhergestellt wird. Benutzt wird immer die Warmstartroutine des zuletzt geladenen Treibers. Diese setzt den Sprungbefehl auf Adresse 5 neu, führt ein Disk-Reset (BDOS-Funktion Nr. 13) durch und kehrt in den CCP zurück.
Der CCP bleibt im Speicher resident, womit nachgeladene Treiber außer ihrem eigenen Speicherbedarf einmalig 2 KByte für den CCP beanspruchen. Auch bei beliebig vielen geladenen Treibern bleibt die SUBMIT-Stapelverarbeitung voll funktionsfähig. Soll XSUB bei geladenen Treibern verwendet werden, so ist ein „Patch" an XSUB erforderlich:
Die Bytes auf den Adressen 158H bis 15FH (dort steht ein Vergleich, ob das BDOS mit einer Adresse XX06H beginnt) sind durch 00H (nop) zu ersetzen. Ein so geändertes XSUB ist sowohl mit geladenen Treibern als auch ohne funktionsfähig.

3.3. Anwendungsbeispiele

Mit Hilfe des beschriebenen Treiberrahmens wurden u. a. folgende Anwendungen realisiert:

- RAM-Floppy mit 16-Bit-Erweiterungsmodul EM 256 oder mit TEST-RAM des Kombinates VEB Elektro-Apparate-Werke Berlin
- Drucker-Zeichen-Umkodierung zur Nutzung des deutschen Typenrades (mit Umlauten und ß) auf dem SD 1152 auch ohne entsprechenden 2. Zeichensatz
- Drucker-Zeichen-Umkodierung zur Simulierung von Umlauten durch Übereinanderdrucken von " und a, o, u, A, O, U für den SD 1157
- „Notizblatt": Jederzeit einblendbares Bildschirmfenster mit Mini-Editor, um z. B. bei der Programmentwicklung und -korrektur Notizen anfertigen und abrufen zu können; als Varianten für A 5120/A 5130 (automatisch an BAB1/2 angepaßt), PC 1715 (unter CP/A automatisch an BAB1/2 angepaßt) und A 5110.

4. Programm DRVUTL

Zum Anzeigen, Aktivieren, Deaktivieren und Entfernen von Treibern wurde in der Programmiersprache PASCAL das Hilfsprogramm DRiVerUTiLity geschrieben. Es zeigt alle geladenen Zusatztreiber, deren Zustand (aktiv/inaktiv, aktivierbar/nicht aktivierbar) sowie den verbleibenden TPA an. Der jeweils zuletzt geladene Treiber kann entfernt werden (Rücktausch der Sprungleisten und Fehleradressen sowie Änderung des Adreßteils des Sprungbefehls auf Adresse 5), alle inaktiven Treiber können aktiviert, alle aktiven Treiber, die deaktivierbar sind, können deaktiviert werden.
DRVUTL wurde auch in Maschinensprache als nachladbares, residentes und durch Eingabe von Escape aktivierbares Programm auf der Basis des Treiberrahmens implementiert /2/.
Mit Hilfe des vorgestellten Treiberrahmens sind schnell und einfach Anpassungen an bestimmte vom Betriebssystem nicht unterstützte Hardwarekomponenten möglich. Weitere denkbare Anwendungen sind z. B. Treiber für Lochband- und Kassettentechnik.

Literatur

/1/ Brode, M.; u. a.: Betriebssystem SCPX. VEB Robotron Büromaschinenwerk Sömmerda 1985
/2/ Wermann, M: Dokumentation „Residentes Programm DRVUTL". IH Mittweida, Sektion Informationselektronik 1987

✉ KONTAKT ☎
Ingenieurhochschule Mittweida,
ORZ, WB Informatik,
Platz der DSF 17,
Mittweida, 9250
Tel. 5 84 69, Koll. Geiler

Ausgabe von Pseudografikzeichen auf Matrixdrucker

Karsten Schiwon
VEB Mikroelektronik „Wilhelm Pieck"
Mühlhausen

Die Kleincomputer KC 85/2 und /3 bieten neben der Vollgrafik auch die Möglichkeit der Verwendung von Pseudografikzeichen. In diesem Beitrag ist eine Möglichkeit aufgezeigt, diese Pseudografikzeichen auf Matrixdruckern abzubilden.

Im Betriebssystem CAOS sind in den Systemzellen 4 Adressen für Zeichenbildtabellen vereinbart, die vom Anwender beliebig verändert werden können. Die Veränderung der Zeichenbildtabellen erfolgt durch den Eintrag einer neuen, vom Anwender zu bestimmenden Zeigeradresse auf die entsprechende Systemzelle des Betriebssystems CAOS. Hierbei ist zu beachten, daß die Zuordnung der Zeichenbilder zu den ASCII-Codierungen in vier Bereiche unterteilt ist. Dabei gelten die in Tafel 1 angegebenen Werte.
Weiterhin gilt, daß für ein Zeichen 8 Bytes in der jeweiligen Zeichenbildtabelle benötigt werden. Die Zuordnung der Pixel zum vereinbarten Zeichen in der Tabelle ist so, daß das erste Byte die oberste Zeile, das zweite die nächstfolgende, ..., das letzte die letzte Zeile vereinbart. Die Bits, deren Bedeutung 0 ist, zeigen keinen Punkt auf dem Bildschirm, die, deren Bedeutung 1 ist, zeigen einen Punkt.
Nach einem RESET oder nach dem Einschalten des Computers werden die Zeigeradressen des Betriebssystems, und somit auch die Zeiger für die Zeichenbilder, in den Grundzustand gebracht.
Um eine Abbildung der Steuercodes, die vereinbarten Steuerzeichen sind von 0 bis 1FH, zu erreichen, ist es weiter notwendig, die Speicherzelle STB (B7A2H), wie in /1/ Seite 75 beschrieben, zu verändern. Eine Zuordnung der im Grundzustand vereinbarten Zeichenbilder zu den ASCII-Codes ist in /2/ dargestellt.
Erfolgt die Ausgabe eines Zeichens über einen USER-Kanal, werden nur die ASCII-Codes übergeben. Ein Drucker ordnet diesen Kodierungen interne Zeichendarstellungen zu, die im ROM des Druckers abgelegt sind. Damit ist der Zeichensatz, der auf diesem Weg darstellbar ist, auf den des jeweiligen Druckers begrenzt. Auf diesem Weg ist die Darstellung der Sonderzeichen des KC 85/3 sowie vom Anwender vereinbarter Zeichen nicht möglich. Aufgrund der Möglichkeit eines Einzelpunktmodus von Matrixdruckern können Sonderzeichen dennoch gedruckt werden. Dies geschieht über die Zuordnung des Sonderzeichens, das im KC aus 8 ∗ 8 Bildpunkten besteht, zu 8 ∗ 8 Druckpunkten des Matrixdruckers. Die Information wird in 8 Bytes (Übertragung von 8 Bit/Zeichen) übergeben. Zur richtigen Interpretation der Daten im Drucker wird vor der eigentlichen Punktinformation eine Steuercode-Folge (ESC-Folge) übertragen. Dadurch wird es möglich, solche Zuordnungen der Zeichenbilder zu den ASCII-Codierungen zur Erleichterung der weiteren Arbeit abzudrucken.
In Tafel 2 ist ein Programm abgebildet, das alle Zeichencodes des KC als Pseudografikzeichen ausgibt. Zur Reduzierung des Aufwandes an Software wurden die Treiberprogramme der Kassette C0171-V24-Software genutzt. Die Erweiterung ist ebenfalls für die in /7/ beschriebene CENTRONICS-Schnittstelle nutzbar.

Tafel 1 Zuordnung der Zeichenbilder zu den ASCII-Codierungen

Systemzelle	Adresse	Zeichencode
CCTL0	B7A6H	20H bis 5FH
CCTL1	B7A8H	00H bis 1FH und 60H bis 7FH
CCTL2	B7AAH	A0H bis DFH
CCTL3	B7ACH	80H bis 9FH und E0H bis FFH

Programmbeschreibung

Das in Tafel 2 aufgeführte Programm ist in die Teile Initialisierung und Ausgabeprogramm unterteilt. Der Initialisierungsteil ist so ge-

Tafel 2 Assemblerlistung der Erweiterung

```
BC00                ;****************************************
BC00                        ORG     0BC00H
BC00                ; ERWEITERUNG FUER V24 TREIBER K.SCHIWON 2.12.1986
BC00                ;****************************************
BC00                ;WORKRAM
BC00                PUFAD   EQU     0B700H
BC00                ;VEREINBARUNGEN
BC00                CCTL0.  EQU     0B7A6H  ;ADRESSEN DER
BC00                CCTL1   EQU     CCTL0+2 ;CAOS-ZEICHEN-
BC00                CCTL2   EQU     CCTL1+2 ;BILDTABELLEN
BC00                CCTL3   EQU     CCTL2+2 ;(CAOS WORKRAM)
BC00                STB     EQU     0B7A2H  ;VPOKE142+2,X
BC00                OUTAB   EQU     0B7B9H  ;ZEIGER UP-NR.0
BC00                UOUT1   EQU     0B7BDH  ;USER OUT 1
BC00                ; SD1 = ADRESSE DER DIREKTEN BYTEAUSGABE
BC00                ; DES TREIBERPROGRAMMS
BC00                SD1     EQU     0BB35H  ;K6311
BC00
BC00 01             MERK    DEFB    1         ; VPOKE ADRESSE = 15360
BC01                ;(MERK)=0--> AUSGABE UEBER NORMALE
BC01                ;               DRUCKERROUTINE
BC01 F1             SPRUNG1 POP     AF
BC02 C3             SPRUNG  DEFB    0C3H      ;JP
BC03 0000           AUSG    DEFW    0         ;ADRESSE AUS USER OUT 2
BC05 7F7F                   DEFW    7F7FH
BC07 56323450               DEFM    'V24PSEUDOGR'
BC12 01                     DEFB    1         ; PROLOG
BC13 CD0ABA         START   CALL    0BA0AH  ; NEUE SELBSTSTARTADRESSE
BC16 21BEB7                 LD      HL,UOUT1+1
BC19 7E                     LD      A,(HL)
BC1A 3203BC                 LD      (AUSG),A
BC1D 23                     INC     HL
BC1E 7E                     LD      A,(HL)
BC1F 3204BC                 LD      (AUSG+1),A
BC22                ; INIT SYSTEM-OUT
BC22 2129BC                 LD      HL,ASCP
BC25 22BEB7                 LD      (UOUT1+1),HL
BC28 C9                     RET
BC29 F5             ASCP    PUSH    AF
BC2A 6F                     LD      L,A
BC2B E5             ASCP1   PUSH    HL
BC2C C1                     POP     BC
BC2D 3A00BC                 LD      A,(MERK)
BC30 A7                     AND     A
BC31 28CE                   JR      Z,SPRUNG1         ;AUSGABE NORMALE
BC33 7D                     LD      A,L     ;NR.ASCII
BC34 FE20                   CP      20H
BC36 3010                   JR      NC,CCT1
BC38                ;STEURZEICHEN AUSGEBEN ODER AUSFUEHREN ?
BC38 3AA2B7                 LD      A,(STB) ;STEUERBYTE CAOS
BC3B CB5F                   BIT     3,A
BC3D 28C2                   JR      Z,SPRUNG1         ;UEBER NORMALE


BC3F                ; ALS PSEUDOGRAFIK
BC3F 7D                     LD      A,L
BC40 ED5BA8B7               LD      DE,(CCTL1)
BC44 0600                   LD      B,0
BC46 1836                   JR      GET
BC48 FE60           CCT1    CP      60H
BC4A 3008                   JR      NC,CCT2
BC4C ED5BA6B7               LD      DE,(CCTL0)
BC50 0620                   LD      B,20H
BC52 182A                   JR      GET
BC54 FE80           CCT2    CP      80H
BC56 3008                   JR      NC,CCT21
BC58 ED5BA8B7               LD      DE,(CCTL1)
BC5C 0640                   LD      B,40H
BC5E 181E                   JR      GET
BC60 FEA0           CCT21   CP      0A0H
BC62 3008                   JR      NC,CCT3
BC64 ED5BACB7               LD      DE,(CCTL3)
BC68 0680                   LD      B,080H
BC6A 1812                   JR      GET
BC6C FEE0           CCT3    CP      0E0H
BC6E 3008                   JR      NC,CCT4
BC70 ED5BAAB7               LD      DE,(CCTL2)
BC74 06A0                   LD      B,0A0H
BC76 1806                   JR      GET
BC78 ED5BACB7 CCT4          LD      DE,(CCTL3)
BC7C 06C0                   LD      B,0C0H
BC7E EB             GET     EX      DE,HL   ;HL-ANFANG
BC7F 110800                 LD      DE,8
BC82                ;BERECHNUNG DER Z-ADRESSE
BC82                ;ZEICHEN IN A OFFSET B
BC82 90                     SUB     B
BC83 2804                   JR      Z,ZGET
BC85 19             BERZ    ADD     HL,DE
BC86 3D                     DEC     A
BC87 20FC                   JR      NZ,BERZ
BC89 1100B7         ZGET    LD      DE,PUFAD
BC8C 010800                 LD      BC,8
BC8F EDB0                   LDIR                      ;ZEICHEN IN PUFR
BC91 CDB4BC                 CALL    INBIMO
BC94 2E80                   LD      L,80H
BC96 1100B7         NEXBIT  LD      DE,PUFAD
BC99 0600                   LD      B,0
BC9B 1A             NEXTBI  LD      A,(DE)
BC9C A5                     AND     L
BC9D 2801                   JR      Z,NOCF
BC9F 37                     SCF
BCA0 CB10           NOCF    RL      B
BCA2 13                     INC     DE
BCA3 3E08                   LD      A,8
BCA5 BB                     CP      E
BCA6 20F3                   JR      NZ,NEXTBI
BCA8 78                     LD      A,B
BCA9 CD35BB                 CALL    SD1
BCAC CB0D                   RRC     L


BCAE CB7D                   BIT     7,L
BCB0 28E4                   JR      Z,NEXBIT
BCB2 F1                     POP     AF
BCB3 C9                     RET
BCB4                ;BITMODE INIT
BCB4 21C3BC         INBIMO  LD      HL,IIT03
BCB7 0604                   LD      B,IITE-IIT03
BCB9 C5             IMBI    PUSH    BC        ;B-COUNTER
BCBA 7E                     LD      A,(HL)
BCBB 23                     INC     HL
BCBC CD35BB                 CALL    SD1
BCBF C1                     POP     BC
BCC0 10F7                   DJNZ    IMBI
BCC2 C9                     RET
BCC3                ; TABELLE FUER STEUERZEICHEN
BCC3 1B             IIT03   DEFB    1BH
BCC4 4B                     DEFB    4BH
BCC5 08                     DEFB    8
BCC6 00                     DEFB    0
BCC7                IITE    EQU     $

ERRORS: 0000
```

Tafel 3 Beispiel zur Benutzung der veränderten Treiberroutine

```
10 '*************************************************
20 ' BEISPIEL ZUR BENUTZUNG DER ERWEITERTEN DRUCKERROUTINE
25 ' GETESTET MIT K6312
30 ' KARSTEN SCHIWON 1987
40 '*************************************************
50 STB=14242:' STEUERBYTE ZUM DRUCK VON STEUERCODES
60 MER=15360:' Umschaltung zwischen den Treiberprogrammen
70 VPOKEMER,0:' normale Druckerausgabe
80 PRINT#2" Uebersicht der vorhandenen Zeichenbilder"
90 PRINT#2:PRINT#2" Zeichencode hexadezimal":PRINT#2
100 ' TABELLENKOPF
110 PRINT#2"    low I ";
120 Z1=VPEEK(14246):Z2=VPEEK(14247):VPOKE14246,0:VPOKE14247,238
130 ' KC ZEICHENSATZ EIN
150 VPOKEMER,1:FORX=0TO15
160 READZA$:PRINT#2" ";ZA$;" ";
170 NEXT:VPOKEMER,0:RESTORE
180 PRINT#2
190 VPOKE14246,Z1:VPOKE14247,Z2:' ANWENDERZEICHENSATZ EIN
200 PRINT#2"   HIGH I":PRINT#2STRING$(62,"-")
210 A$="    I":' linker Rand
220 NULL0:' Unterdruecken nachfolgende Nullen
230 FORY=0TO15:' ZEILEN
240 READY$:GOSUB280
250 NEXT
260 END
270 DATA"0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F"
280 ' UNTERPROGRAMM EINE ZEILE, ZEILENNUMER MIT Y UEBERGEBEN
290 PRINT#2"   ";Y$;A$;
300 VPOKEMER,1:' PSEUDOGRAFIK
310 VPOKESTB,8:' AUSGABE STEUERZEICHEN
320 FOR X=0 TO 15:PRINT#2" ";CHR$(16*Y+X);" ";:NEXT
330 VPOKEMER,0: VPOKESTB,0:PRINT#2:RETURN
```

staltet, daß die vorhandenen Druckerroutinen der Kassette C 0171 benutzt werden können. Das Programm V24PSEUDOGR initialisiert in der abgedruckten Form den USER-Ausgabekanal 1 (LIST#2) des Systems. Soll der Ausgabekanal 2 des Systems benutzt werden, so muß anstelle der Vereinbarung UOUT1 die Adresse UOUT2 des Systems angegeben werden. Es ist nur ein USER-Kanal zugelassen. Dieser muß mit dem USER-Kanal, der in der Treiberroutine des Druckers initialisiert wurde, übereinstimmen. Anstelle der Ausgabeadresse des Treiberprogramms von der Kassette wird die Adresse von ASCP bei Aufruf von V24-PSEUDOGR in die Systemzelle eingetragen. Um die Arbeit mit dem Treiberprogramm zu erleichtern, kann der in Tafel 2 dargestellte Teil mit dem Treiberprogramm der Kassette auf Magnetband aufgezeichnet werden. Hierbei verändern sich die Endadresse und die Startadresse des Treiberprogramms.

Achtung: Bei Benutzung der veränderten Treiberroutine ist eine gleichzeitige Nutzung der Ein-/Ausgangsroutinen V24IN2 und V24DUPL2 der Kassette nicht möglich.

Der Ausgabeteil kann über 2 RAM-Zellen gesteuert werden. Die erste RAM-Zelle ist die auf der Adresse 0BC00H vereinbarte Merkzelle, die die Zeichenausgabe zwischen den vorhandenen Treiberprogrammen, normale Druckerausgabe oder Druck der Pseudografikzeichen, regelt. Ist der Inhalt dieser Zelle 0, so wird der Gerätetreiber der Kassette C0171 benutzt. Ist der Inhalt der Zelle ungleich 0, so werden die Zeichen als Pseudografik ausgegeben. Um die Zeichenbilder der Steuerzeichen zu erreichen, ist es notwendig, das Steuerbyte STB im CAOS-Work-RAM zu verändern. Hierzu ist in Tafel 3 ein Beispielprogramm in BASIC angegeben, das eine vollständige Übersicht der benutzbaren Zeichenbilder ausdruckt. Wird die Steuerung an die abgebildete Treiberroutine übergeben, erfolgt als erstes die Berechnung der Zeichenadresse, danach werden ab der berechneten Adresse 8 Bytes in den I/O-Puffer des Systems übernommen. Diese 8 Bytes entsprechen der auf dem Bildschirm dargestellten Information. Es erfolgt die Initialisierung des Druckers zur Ausgabe von 8∗8 Punkten im Einzelpunktmodus. Die Steuercodefolgen für die einzelnen Drucker können einige Unterschiede aufweisen, deshalb sollten für die entsprechenden Drucker die notwendigen Steuercodefolgen in die Tabelle für Steuerzeichen am Ende des Programms in Tafel 2 eingetragen werden. Für die Drucker K6304, K6311, K6312, K6313, K6314 gilt die dort eingetragene Steuercodefolge.

Da das Ausgabeprogramm Teile der Software der Kassette C0171 nutzt, ist es notwendig, die Adresse der direkten Ausgaberoutine, die bei den Treiberprogrammen unterschiedlich ist, zu beachten. Diese Adresse, die im Programm SD1 vereinbart ist, ist für jedes Treiberprogramm der Kassette C0171 in Tafel 4 aufgeführt.

Tafel 4 Adressen der direkten Ausgaberoutine der Druckertreiber (Kassette C0171)

Treiberprogramm	Adresse für SD1
V24K6304	BB23H
V24K6311	BB35H
V24K6312	BB30H
V24K6313	BB2DH
V24K6314	BB2DH

Literatur

/1/ KC 85/3 System-Handbuch. VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen

/2/ KC 85/3 Übersichten. VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen

/3/ KC 85-Beschreibung zu C0171-V24-Software. VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen

/4/ Manual K6311, K6312. VEB Robotron Büromaschinenwerk Sömmerda

/5/ Manual K6313, K6314. VEB Robotron Büromaschinenwerk Sömmerda

/6/ Manual K6304. VEB Robotron Büromaschinenwerk Sömmerda

/7/ K.-D. Kirves, B. Schenk; K. Schiwon: Digital Ein-/Ausgabemodul KC 85/2 und /3. Mikroprozessortechnik 1 (1987) 8

Superschneller Schaltkreis

Einen neuartigen superschnellen Schaltkreis hat die französische Firma Thomson entwickelt. Als Halbleiter dienen Einkristalle, bestehend aus einem Atomgitter von Arsen, Gallium und Indium. Thomson gehörte schon vor einigen Jahren zu den ersten Unternehmen, die Arsen-Gallium-Halbleiter entwickelten, der eine etwa sechsmal höhere Geschwindigkeit des Elektronenflusses gegenüber herkömmlichen Silizium-Halbleitern aufwies. Mit der Entwicklung des Arsen-Gallium-Indium-Halbleiters läßt sich die Mobilität der Elektronen weiter erhöhen. *ADN*

Subroutinen des BASIC-Interpreters von den KC-Rechnern

1

```
                         KC 85/2-BASIC ROMPAC
                         ====================

C000        Basic-Neustart
----        --------------
C005 Prolog 7F7FH 'BASIC' Initialisierung des Notizspeichers durch
     Blocktransport (C0BD-C120===>300-363) / Reservierung Stack

C08C        INTERPRETER - WARMSTART
----        -----------------------
C082 Prolog 7F7FH 'REBASIC'

C092   Strings (HC-BASIC,BYTES FREE,MEMORY END)

C0BD NOTIZSPEICHERERSTBELEGUNG

C121        LISTE DER RESERVIERTEN WORTE
----        ----------------------------
  (BASIC Token / Abschluß mit 80H/ +80H am Wortbeginn)
END,FOR,NEXT,DATA,INPUT,DIM,READ,LET,GOTO,RUN,IF,RESTORE,GOSUB,RETURN,REM,STOP,
OUT,ON,NULL,WAIT,DEF,POKE,DOKE,AUTO,LINES,CLS,WIDTH,BYE,!,CALL,PRINT,CONT,LIST,
CLEAR,CLOAD,CSAVE,NEW,TAB(,TO,FN,SPC(,THEN,NOT,STEP,+,-,*,/,^,AND,OR,>,=,<,SGN,
INT,ABS,USR,FRE,INP,POS,SQR,RND,LN,EXP,COS,SIN,TAN,ATN,PEEK,DEEK,PI,LEN,STR$,
VAL,ASC,CHR$,LEFT$,RIGHT$,MID$,LOAD,TRON,TROFF,EDIT,ELSE

C240        ADRESSLISTE DER UNTERPROGRAMME
----        ------------------------------

C91A,C7DE,CCDC,CA48,CBEC,CF01,CC1F,CA5D,CA07,C9EB,CACF,C8DF,C9F6,CA25,CA4A,C918,
D3EC,CAB3,CBC0,D3F7,D0C4,D437,D44E,C5FA,C6EA,DDD0,CBB9,DFF4,CA4A,DB38,CAFA,C948,
C6F2,C9AA,DC43,DD41,C640,C62D,C7B7,C7B8,C3E7,CA4A,D6A6,D770,D6BC,0303,D090,D3E3,
D0BD,D91F,D9FD,D559,D96D,DA70,DA76,DAD7,DAEC,D431,D444,D6D5,D32C,D156,D3BF,D33B,
D34B,D35B,D389,D392
(Wichtungskoeffizienten)
(79) D817, (79) D46A, (7C) D598, (7C) D5F3, (7F) D928, (50) CE5E, (46) CE5D

C2DD        FEHLERAUSGABE UND RAMTEST
----        -------------------------
     Fehlerkennzeichen 'NFSNRGODFCOVOMULBSDD/OIDTMOSLSSTCNUFMOIO'
C305 Fehlerkennzeichenstring ' ERROR' / ASCBEL / (00H)
C30D String ' IN ' / (00H)
C312 String 'FILE FOUND' / ASCCR
C31D String 'OK' / ASCCR / (00H)
C321 String 'BREAK' / (00H)

C327        TEST, OB RAM FÜR DIE NäCHSTEN OPERATIONEN AUSREICHT
----        ---------------------------------------------------
Neuer zusätzlicher Rambedarf= 2*(C) / aktivierter Ram und SP müssen mindestens
```

2

```
49 Bytes auseinander liegen, sonst OM-ERROR

C33E        ORGANISATION DER FEHLERAUSGABE
----        ------------------------------
(Fehlernummer in Register E)
   OM ERROR
C342  Legt Zeilennr.des zuletzt gelesenen Datastatements in (358H) ab.
(Vorbereitung für Basic-Ausschrift "?SN ERROR IN xx". (xx=aktuelle Zeilennr.)
C348 SN ERROR
C34B /0 ERROR
C34E NF ERROR
C351 DD ERROR
C354 UF ERROR
```

Fehler	Bedeutung	E = nn
NF	Next Without For	00
SN	Syntax	02
RG	Return Without Gosub	04
OD	Out of Data	06
FC	Illegal Funktion Call	08
OV	Numeric Overflow	0A
OM	Out of Memory	0C
UL	Undefined Line Number	0E
BS	Bad Subscript	10
DD	Double Dimensioned Array	12
/0	Division by Zero	14
ID	Illegal Direct Mode	16
TM	Type Mismatch	18
OS	Out of String Space	1A
LS	String too long	1C
ST	String Formula too complex	1E
CN	Can't Continue	20
UF	Undefined User Function	22
MO	Missing Operand	24

```
Bei Nutzung dieser Routine wird E mit dem Fehlercode geladen

C387        EDITIERROUTINE/KOMMANDOEBENE
----        ----------------------------
C388 Basic-Ausschrift "OK" - Kommandoebene
C396 Zeileneingabe
C42B Zeilennr. prüfen und in Hexzahl wandeln
C42F Eingabezeile in verkürzte Darstellung wandeln
C435 Unterprogrammadresse für Kommando bilden und Kommando
     ausführen, wenn keine Zeilennr. vorhanden ist
C43E Test, ob der Zeilennr. weitere Zeichen folgen
C443 Zeiger (in Register HL) auf Zeile mit der nächstgrößeren
     Zeilennr. stellen
C44D Sprung zur Fehlerausgabe (UL), wenn zu löschende Zeile
     im Programm nicht vorhanden ist
C450 Einsortieren der eingegebenen Zeile in das schon im RAM
```

3

```
     vorhandene Programm
C4AB Zeile einfügen
C4BB Zeiger auf Anfang der Zeile mit der nachstgrößeren Zeilennr.
     stellen / Z-Flag=1, wenn Zeilennr. schon vorhanden

C4DA        UMWANDLUNG DER EINGEGEBENEN ZEICHENKETTE
            IN DIE INTERNE (VERKÜRZTE) DARSTELLUNG
            DIESE IM EINGABEPUFFER ABLEGEN
----        ----------------------------------------
C4E4 Erweiterungslistenflag (3FBH)=0
C50D Test, ob eingegebene Zeichenkette in der Liste der reservierten
     Worte enthalten ist und merke Stelle in der Liste
C516 Umwandlung eingegebener Klein- in Großbuchstaben
C526 Stelle Zeiger auf nächsten Wortanfang
C53B Erweiterungslistenflag wird gesetzt
C54E Sprung, wenn Wort vollständig identifiziert

C5A0        EINGABE EINER ZEILE UND ABLAGE IM EINGABEPUFFER
            (BERÜCKSICHTIGUNG DER STEUER- UND SONDERZEICHEN)
----        ------------------------------------------------

C5FA AUTO
C62D LOAD
C640 NEW Programmendezeichen an den Anfang des Programmablagerambereichs
     rücken und Notizspeicher löschen
C669 Stackpointer auf obere Ramgrenze stellen
     obere Stackbegrenzung eintragen

C689 VERGLEICHE INHALT VON REGISTER DE MIT HL
Z-Flag gesetzt, wenn DE = HL
C-Flag gesetzt, wenn DE < HL

C68F TEST AUF GROSSBUCHSTABEN
C-Flag gesetzt, wenn Kleinbuchstaben

C697 TEST AUF ZULäSSIGE IO-OPERATION (DATENSCHUTZ)
  Wenn (35E) > 0, dann Ausschrift ' ERROR'            862 D

C6A2 Cursor links
C6A9 AUSGABE EINES LEERZEICHENS

C6AE ZEICHEN ZUR AUSGABESCHNITTSTELLE, ASCII IN REGISTER A
 C6B3 Ausgabeunterdrückung, wenn Steuerzeichen
 C6C4 Zeiger auf nächstes Zeichen

C6CF EINGABEANFORDERUNG STELLEN
Diese  Routine  realisiert  eine  Input-Funktion.  Der  Sprungbefehl  für  das
Eingabeprgramm steht auf 034AH. (Standard: C3AEC5)
```

4

```
     Ausgabe "?"
C6D4 Cursor rechts/ Sprung ==>INPUT
C6DF Initialisierung Eingabepuffer

C6EA LINES  Zeilenzahl in 0346H
C6F2 LIST
C6B7 TRON   Trace-Flag (030AH)= AFH
C7B8 TROFF  Trace-Flag (030AH)= 00H

C7BD ABLAUFSTEUERUNG INTERPRETIERVORGANG

C7DE FOR    For-Flag (03CCH)= 64H

C854        RUN-MODE ORGANISATION
----        ---------------------
Der  Basic-Interpreter  startet  hier. Es wird ein Test auf Unterbrechungsanfor-
derug  durchgeführt  und  die  Adresse  des  akt.  Statements  (HL)  auf  03CFH
gespeichert.
C856 Test auf Fortsetzung der Zeile mit Colon (:)
C863 ?SN ERROR, wenn Colon fehlt
C866 Test auf Basic-Ende (00 00 00)
C875 Test auf Trace-Funktion / Ausführung
C88A Zurück zu Run-Mode, wenn Zeilenende

C892        ERMITTLUNG DER ZU DEN KOMMANDOS GEHÖRENDEN UP-ADRESSEN
----        ------------------------------------------------------
            (Direct-Mode / auch im Indirect-Mode verwendbar)
PA:Reg  A  =  Token  (HL)  /  zulässig  80H-A4Hund  D0H-D4H,  sowie  D5H-F4H aus
Erweiterung
C894 Sprung zu  LET  wenn  80H
C8D9 ?SN ERROR, wenn A4H<Token<D0 (Direct Mode nicht möglich!)
C8A1 Wenn C-Flag gesetzt, dann Standardtoken
C8A8 Token-Erweiterung / wenn Z-Flag ( 03FCH =0), dann ?SN ERROR
C8AA Absprung in Erweiterung (0E003H)
C8BB Unterprogrammadrese aus BC in den Stack  (Aufruf nach RET)
C8BD Zeiger in aktueller Zeile auf nächstes signifikantes Zeichen
C8BF Test auf Zahl (Rücksprung bei CHR "9")
C8C2 Test auf Zeilenende
C8C6 Test auf Zahl (Carry gesetzt, wenn Ziffer)

C8CC VERGLEICH (HL) MIT ((SP))
 C8CF (SP) wird um 1 erhöht
 C8D1 Bei Gleichheit Sprung zu C8BD, sonst ?SN ERROR
 C8D6 Vergleich (HL) mit "," / Sprung zu C8D1
 C8DB Vergleich (HL) MIT ")" / Sprung zu C8D1

C8DF RESTORE
 C8E9 Zeilennummer (Hexadezimal) in DE
 C8F0 Wenn Zeile nicht vorhanden, dann ?UL ERROR
```

5

```
C8F9        PROGRAMMUNTERBRECHUNG UND -FORTSETZUNG
----        --------------------------------------
C8FD Test auf Stoptaste  (ASC=13H)
C901 Test auf Breaktaste (ASC=3)
C90C Test auf Conttaste  (ASC=1EH)
C90F Test auf Starttaste (ASC=0AH)


C918 STOP
C91A END
 C93F 'BREAK' im Direct-Mode
      'BREAK IN' Zeilennummer im Indirect-Mode
C948 CONT C94D  CN ERROR


C95B BESTIMMEN DES WERTES EINES AUSDRUCKS
     Zeiger in HL / Ergebnis in (SP) /  C967 FC ERROR


C986 UMWANDLUNG EINGEGEBENER ZAHLEN IM ASCII IN HEXADEZIMALZAHLEN
     Zeiger in HL  Ergebnis in DE


C9AA CLEAR
C9EB RUN
C9F6 GOSUB
CA07 GOTO CA20: UL ERROR
CA25 RETURN CA2E: RG ERROR
CA48 DATA
CA4A ELSE und REM
CA5D LET
CAB3 ON
CACF IF
CAFA PRINT
 CB1B Abfrage nach "," (formatierte Ausgabe)
 CB1F Abfrabe nach ";" (keine Ausgabe von CR und LF)
 CB2F Ausgabe einer Zahl, wenn (3AE)=0
 CB4F Ausgabe eines Strings
 CB55 Ausgabe eines LF- und CR-ASCII am Zeilenende, zusätzlich Ausgabe
      der vereinbarten Anzahl von Dummyzeichen
    ((340)<- Anzahl der Dummyzeichen)
 CB7C Organisation der Ausgabeformatierung bei mehreren Listenelementen
      in einer PRINT-Anweisung
 CB92 Organisation des Tabulatorsprunges

CBB9 WIDTH
CBC0 NULL


CBC9 String '?REDO FROM START' / 0DH
CBD8 Ausgabe ?REDO FROM START, wenn (3CE)=0, sonst ?SN ERROR


CBEC INPUT
```

6

```
CC0C Eingabeanforderung
CC1F READ
CC3E Wiederholung der Eingabeanforderung
CCA3 Ausgabe ?EXTRA IGNORED, wenn Z-Flag rückgesetzt
CCAB String '?EXTRA IGNORED' / 00H
CCC5 OD ERROR


CCDC NEXT / VORRANGSTEUERUNG DER OPERATIONEN


 CD16        AUFSTELLUNG EINES RPN-VEKTORS IM STACK
             UM VERKETTETE AUFGABEN ZU LÖSEN
 ----        ----------------------------------------
Zeiger auf erstes Zeichen in HL und Abfrage nach:
CD1B   ,
CD31 TM ERROR
CD36  (
CD84 MO ERROR
CDC1  +
CDC5  .
CDCA  -
CDCE  "
CDD3  NOT
CDD8  FN
CDDD  Funktionen ( SGN bis  MID$ )
CE09  Berechnung der Sprungadresse, wenn (HL)=  SGN  bis  MID$


CE5D OR
CE5E AND
CEE3 NOT
CF01 DIM  D00B BS ERROR
D090 FRE
D0BD POS
D0C4 DEF
D0F0 FN


D137 TEST, OB PROGRAMMABARBEITUNG IM DIRECT MODE / D140 ID ERROR


D156 STR$  D1C3: ST ERROR

D1C9 AUSGABE EINER ASCII-ZEICHENKETTE AUS DEM SPEICHER BIS 0
     Anfangsadresse in HL


D1E1 STRINGARITHMETIK / D1FF OS ERROR / D2C7 LS ERROR


D32C LEN
D33B ASC
D34B CHR$
D35B LEFT$
D389 RIGHT$
```

7

```
D392 MID$
D3BF VAL

D3DB Sprung zur Abfrage nach ")"


D3E3 INP
D3EC OUT
D431 PEEK
D437 POKE
D444 DEEK
D44E DOKE


ARITHMETIKMODUL
===============


  D45E        ADDITION UND SUBTRAKTION
  ----        ------------------------
D45E (03E5)=(03E5)+.5
D461 (03E5)=(03E5)+(HL)
D466 (03E5)=(HL)-(03E5)
D46A (03E5)=(SP)-(03E5)
D46C (03E5)=(SP)+(03E5)
D4CF (03E5)=0
D507 Sprung zu OV ERROR
D516 (03E5)=-(03E5)


 D548 KONSTANTENLISTE FÜR LN-FUNKTION
D559 LN
 D591 (03E5)=(03E5)*LN(2)

  D598        MULTIPLIKATION UND DIVISION
  ----        ---------------------------
D598 (03E5)=(SP)*(03E5)
D59A (03E5)=BCDE*(03E5)
D5F3 (03E5)=(SP)/(03E5)
D5F5 (03E5)=BCDE/(03E5)
D653 OV ERROR


D65B HILFSROUTINEN FÜR ARITHMETIK
 D697 VORZEICHENTEST


D6A6 SGN
D6BC ABS

 D6C8 (03E5) IN STACK (GPZ, 4 BYTES)

D6D5 PI  IN (03E5) (GPZ, 4 BYTES)


 D6DD (HL) NACH (03E5) KOPIEREN
```

8

```
D6E0 BCDE nach (03E5)
D6EB BCDE MIT (03E5) LADEN
D6EE BCDE mit (HL) laden
D6F5 HL=HL+1
D6F7 (03E5) NACH (HL)
D6FA (DE) NACH (HL)
D6FC (DE) NACH (HL) / LÄNGE IN B
D703 VORZEICHENROUTINEN (03E5)=-(03E5)
D71B Vergleich (03E5) MIT BCDE
 wenn größer A=01
 wenn gleich A=00
 wenn kleiner A=FF (-01)
D732 VERGLEICH BCDE MIT (HL)
 bei gleich SP=SP-2

D770 INT


  D7A1        UMWANDLUNG EINGEGEBENER ZAHLEN IM ASCII
  ----        ----------------------------------------
 Abfrage nach:
D7A1 -
D7A6 +
D7B8 .
D7BC E


D821 FORMATISIERE ZAHLENAUSGABE
  ----       -------------------------
D821 Ausgabe des Strings ' IN '
D834 HL=03EA (Print-Puffer)
D83B (HL)=' ', wenn Zahl negativ (HL)='-'
D842 HL=HL+1 / (HL)='0'


D904 KONSTANTE FÜR SQR-FUNKTION / 0.5

D908 KONSTANTENLISTE FÜR KONVERTIERUNG VON ZAHLEN INS AUSGABEFORMAT
GPZ, 3 Bytes
D908 100000
D90B  10000
D90E   1000
D911    100
D914     10
D917      1


D91F SQR  (03E5)=SQR(03E5)
 D928  (03E5)=(SP)^(03E5)
D96D EXP  (03E5)=EXP(03E5)
 D9AD Konstantenliste für EXP

D9CE Routine zur Entwicklung von Funktionen in Reihen
     HL=Zeiger (Anzahl der Glieder)
```

```
D9FD RND
 DA64 Konstantenliste
DA70 COS  (03E5)=COS(03E5)
DA76 SIN  (03E5)=SIN(03E5)
 DABA Konstanten für COS aus SIN und PI/2
 DABE Konstanten für SIN
DAD7 TAN  (03E5)=TAN(03E5)
DAEC ATN  (03E5)=ATN(03E5)
 DB13 Konstantenliste für ATN
DB38 CALL

DB49: Umwandlung hexadezimaler Zahlen im ASCII in interne Darstellung
      HL=Zeiger (erster Code=AE='*'+80H) / Umgewandelte Zahl in DE


DB7B KASSETTEN EIN-/AUSGABEROUTINEN

DC00 CLOAD*  DC3F String 'BAD'
DC47 CLOAD   DC82 IO ERROR
DD45 CSAVE
DD69 CSAVE*
DD00 CLS

DDD5 Ausgabe eines Zeichens A=ASCII
DDE4 Eingabe eines Zeichens A=ASCII
DDF3 Test auf Unterbrechungsanforderung
DDFD Alle benutzten Kanäle schließen
DE25 Aktivieren der I/O-Kanäle / Ausgabe
DE5F Eingabe


   DE98        ZEILENEDITOR
   ----        ------------
DF0F Ende nach CR
DF27 Ende nach STOP
DF63 Zeichen eintragen
DF7A Cursor left
DF85 DEL
DF99 Space
DFC5 Cursor links
DFCF Cursor rechts
DFDB Enter


DFF4 Rücksprung ins CAOS


***** ERWEITERUNG *****
=======================

E571 Token-Tabelle

INKEY$,JOYST,STING$,INSTR,RENUMBER,DELETE,PAUSE,BEEP,WINDOW,BORDER,INK,PAPER,AT,
COLOR,SOUND,PSET,PRESET,BLOAD,VPEEK,VPOKE,LOCATE,KEYLIST,KEY,SWITCH,PTEST,CLOSE,
OPEN,RANDOMIZE,VGET$,LINE,CIRCLE,CSRLIN
E61A: Abschluß mit 80H
```

9

```
E61B Sprungtabelle

E887,02FD,E710,EC2E,E76F,E94F,E974,E9A8,E9BD,EA93,EB04,EB1E,C348,EB3A,EBA2,EBE5,
EC16,EA47,C348,EA76,E853,ECB3,ECA4,ECB7,C348,ED03,ED44,ED3E,C348,ED79,ED7E

E659 Erweiterung 1

E:  <B>  : TOKEN-LDTOK / <HL> : auf TOKEN
A: <<SP>> : Startadresse Routine / <HL> : auf TOKEN


E66D Erweiterung 2  (PRINT)

E: <HL>
auf signif. Zeichen nach PRTOK
wenn (3FDH)<>0, dann SN ERROR
A: <HL> auf Zeilenende
C-Flag gesetzt: INK
C-Flag n.  "  : keine Erw. PRINT INK bis PRINT COLOR
Z-Flag gesetzt: PRINT AT (CALL EAA1H)
 E679 doppeltes Auftreten abblocken
 E68A Farbe retten (3FEH)
 E68D Abfrage INK;COLOR;PAPER
 E6CD CALL PRINT


E6E5 Erweiterung 3

    Verteiler zu zusätzlichen Standardfunktionen
E: <HL> auf signif. Zeichen nach TOKEN
   <BC>=<<SP>> (TOKEN-SGTOK)*2
A: <HL> auf signif. Zeichen nach TOKEN


E710 STRING$
E76F RENUMBER
E94F DELETE
E974 PAUSE
E9A8 BEEP
E9BD WINDOW
 EA27 Standardfenster / Cursor löschen /BWS abschalten
EA47 BLOAD
EA55 VPEEK
EA76 VPOKE
EA93 BORDER
EAA1 AT
EB04 INK
EB1E PAPER
EB3A COLOR
E853 LOCATE
E887 INKEY$
EBA2 SOUND
EBE5 PSET
EC2E INSTR
ECA4 KEY
```

10

```
ECB3 KEYLIST
ECB7 SWITCH
ECCE PTEST
ED03 CLOSE
ED3E RANDOMIZE
ED44 OPEN
ED79 LINE
ED7E CIRCLE
EDAC CSRLIN
EDCF VGET$

                ENDE
                ====
```

11

Erläuternde Bemerkungen gibt unser Autor Prof. Dr. Horst Völz in seinem Beitrag „Universelle Nutzung des BASIC-Interpreters" in MP 7/87.

Effektives Programmieren in BASIC

Prof. Dr. Horst Völz, Berlin

BASIC ist eine sehr beliebte Programmiersprache für Anfänger. Sie zeichnet sich durch leichte Erlernbarkeit und hohe Interaktivität aus und existiert für praktisch alle Mikrorechner. Sie hat aber andererseits Nachteile, wenn effiziente, praktisch nutzbare Programme realisiert werden sollen. Die entsprechenden Probleme betreffen vor allem zwei Punkte: die relativ geringe Abarbeitungsgeschwindigkeit und Unzulänglichkeiten bezüglich des strukturierten Programmierens. Diese Schwierigkeiten lassen sich nun erheblich reduzieren, wenn man einige Eigenarten des üblichen Interpreters berücksichtigt. Dies soll im folgenden aufgezeigt werden.

1. Programmstruktur

Das BASIC-Programm wird nach dem Prinzip von Bild 1 im Programmspeicher abgelegt. Jede Zeile beginnt also mit einem 2-Byte-Pointer, welcher die Adresse zum Pointer der nächsten Zeile enthält. Auf diese Art sind die BASIC-Zeilen untereinander verkettet. Dadurch lassen sich leicht BASIC-Zeilen einfügen. Das Programm endet mit dem Pointer 00 00.
Hinter dem 2-Byte-Pointer folgen weitere 2 Bytes für die Zeilen-Nr. Sie ist binär codiert. Deshalb sind die Zeilen 0 bis 65535 möglich.
Erst nach diesen vier Bytes beginnt der eigentliche BASIC-Text. In ihm sind alle reservierten Wörter (also z. B. FOR, TO, STEP, RUN) als sogenannte Tokens abgelegt. Das Token ist ein Byte, welches größer als 7FH bzw. 127D ist. Nach dieser Methode wird erheblich Speicherplatz gespart und sogar ein Geschwindigkeitsvorteil erreicht. Allerdings müssen dafür die ASCII-Eingaben für die Übergabe in den Programmspeicher erst in Tokens umgerechnet werden. Doch dies erfolgt nur einmal bei der Eingabe. Desgleichen erfolgt die Rückrechnung bei LIST. Jede Programmzeile wird mit 00 abgeschlossen.

2. Bezeichnung der Variablen

Bei den meisten BASIC-Dialekten werden für die Variablenkennzeichnung 2 Bytes verwendet. Sie werden durch die Eingabe eines Großbuchstabens und eines eventuell nachfolgenden zweiten Großbuchstabens bzw. einer Ziffer erzeugt. Werden weitere Buchstaben verwendet, so erscheinen sie zwar im Programmspeicher, haben sonst aber keine Wirkung. Neben den Vorteilen eines für den Menschen leichter lesbaren Programms haben sie auch den Nachteil, daß in ihnen keine – auch nicht abschnittsweise – reservierten Wörter vorkommen dürfen. KONTO

wäre also eine völlig ungeeignete Variablenbezeichnung, enthält sie doch die Wörter ON und TO. Wie die Variablen in den dafür vorgesehenen Speicher abgelegt werden, zeigt Bild 2. Hinter den zwei Bytes zur Kennzeichnung des Namens stehen 4 Bytes, welche den aktuellen Wert der Variablen in einer speziellen Kodierung für binäre Gleitkommazahlen enthalten. Andere Dialekte verwalten auch mehrere Bytes. Bei den String-Variablen ist die Nutzung der 6 Bytes modifiziert:

- Das erste Bezeichnungsbyte ist gegenüber den numerischen Variablen um 80H erhöht.
- Da die Länge der Stringvariablen beliebige Werte zwischen Null und 255D annehmen kann, benötigen sie eine besondere Behandlung. An Stelle der 4 Bytes der numerischen Variablen, stehen daher:

○ Das erste Byte weist die Länge des String aus.
○ Das zweite Byte ist 00 (in gewissen Dialekten auch FFH).
○ Die beiden letzten Bytes sind die Adresse für den Beginn des Ortes, wo das String abgelegt wurde.

Die vier Bytes sind hier also nur Hilfsgrößen. Die Strings selbst sind anderweitig abgelegt. In einigen Dialekten gibt es noch weitere Variablen, die dann wiederum speziell gekennzeichnet sind, z. B. % für Integer-Variablen (ganze Zahlen) und ! für doppelt-genaue Variablen. Sehr große BASIC-Interpreter lassen schließlich noch relativ lange Namen für Variablen zu. Auf all diese Besonderheiten sei im folgenden nicht weiter eingegangen.

3. Besonderheiten bei Arrays

Felder (arrays) werden sehr ähnlich wie die Variablen behandelt. Sie werden in einem speziellen Speicherbereich abgelegt und benötigen nur einmal für jedes Feld die 2 Bytes zur Bezeichnung. Hieran schließen sich 2 Bytes an, welche die Gesamtanzahl von Bytes für die Datenstruktur (Länge der danach folgenden Bytes des Feldes) angeben. Dann folgt ein Byte, welches die Anzahl der Dimension enthält. DIM (1, 2, 3) legt hier also eine 3 ab. Je nach der Anzahl der Dimensionen folgen je 2 Bytes für die Laufzahl in jeder Dimension. Eine Dimension kann also theoretisch bis zu 65535 Elemente besitzen. Dann erst folgen dicht aneinander gefügt die einzelnen je 4 Bytes eines Array-Elementes. Hier existiert wieder die Struktur ähnlich wie bei den Variablen, und zwar je nachdem ob numerische oder String-Arrays vorliegen. Bild 3 faßt diese Aussagen an einem Beispiel zusammen.

```
!-----------------------------------!
! Name !  Byte  ! Anschließende Daten !
!-----------------------------------!
!  A   ! 00 41  !   Gleitkommazahl    !
!  A1  ! 31 41  !         im          !
!  AB  ! 42 41  !  4  Byte Binärcode  !
!-----------------------------------!
!  A$  ! 80 41  !   1 Byte für Länge  !
!  A1$ ! B1 41  ! 00 ! 2 Byte Adresse !
!-----------------------------------!
```

Bild 2 Bezeichnung der Variablen. Bei Strings ist das erste Namens-Byte um 80H erhöht. Außerdem enthält das folgende 4-Byte-Feld nur Hilfsgrößen.

```
Die Belegung ist beim KC 85/1 und KC 87 sehr ähnlich.
Die Belegung beim Bandinterpreter hat folgende Änderungen.
RAM für Bandinterpreter liegt von 200 bis 2A00H.
Der Workspace liegt statt 300 bis 3FFH bei 2A00 bis 2AFFH.
Dem entspechend sind auch die Parameter verschoben,
also z.B. 3DBH -> 2ADBH

        Hauptspeicher                               Workspace
        =============                               =========
Adresse  Bedeutung    Pointer                  Adresse  Bedeutung

Ende    !-----------!      CLEAR A,B    --> 3FF    !----------!
(3FFF)  !  freier   !                  !           !Farb-Byte !
        !    RAM    !                  !    3FE    !----------!
        !-----------! (3B0) <------    !           !Arithmetik!
        !  String   !              !   !    3EA    !----------!
        !  belegt   !              !   !           !PRINT-REG.!
        !-----------! (3C4) CLEAR A,B  !    3E9    !----------!
        !  String   !             !    !           !Arithnetik!
        !   frei    !             !    !    3E1    !----------!
        !-----------! (356) <----      !           !          !
        !   Stack   !                  !    3C4    !----------!
        !     !     !                  !           !  String- !
        !     V     !                  !           ! tabellen !
        !...........!                  !    3B4    !----------!
        !           !                  !           !          !
        ! dynamisch !                  !    3AC    !----------!
        !  bedingt  !                  !           !  Cursor  !
        !   frei    !                  !    3AB    !----------!
        !-----------! (3DB)            !           ! Eingabe- !
        !  Arrays   !                  !           !  buffer  !
        !-----------! (3D9)            !    361    !----------!
        ! Variablen !                  !           !Programm- !
        !-----------! (3D7)            !           !  start   !
        !  BASIC-   !                  !    35F    !----------!
        ! Programm  !                  !           !          !
 401    !-----------! (35F) <---------      33C    !----------!
        !           !                              ! Zufalls- !
        ! Workspace !                              ! generator!
        !           !                       319    !----------!
 300    !-----------! <-----------                 !          !
        !           !            !          306    !----------!
 200    !-----------!            !                 ! Sprünge  !
        ! Workspace !            !                 ! USR-Fkt. !
        !   CAOS    !            !                 !Warmstart !
 140    !-----------!             ------->  300    !----------!
        !  freier   !
        !   RAM     !
  0     !-----------!
```

Bild 4 Speicherbelegung beim KC 85/2 und KC 85/3.

4. Anordnung im Speicherraum

Die Aufteilung des Speichers wird hier am Beispiel des ROM-Interpreters von KC 85/2 und KC 85/3 behandelt. Sie stimmt in den wichtigsten Fakten mit dem des KC 85/1 und des KC 87 überein. Für die Bandversionen sind additive Verschiebungen im Speicherraum erforderlich. Fast alle BASIC-Interpreter verwenden das gleiche Prinzip, weichen jedoch in den absoluten Adressen ab. Das BASIC-Programm wird ab einer Anfangsadresse abgelegt. Davor steht eine 00. Diese Adresse wird im Arbeitsspeicher (hier 300 bis 3FFH) auf einer festen Adresse (hier 35FH) abgelegt. Durch Änderung dieser Adresse sind folglich BASIC-Programme auch verschiebbar. Am Ende des Programmes liegt der Speicherraum für die einfachen Variablen. Auch sein Beginn und Ende wird durch je einen Pointer (3D7H und 3D9H) festgehalten. An die Variablen schließen sich die Arrays an. Ihr Ende enthält ein weiterer Pointer 3DBH. Es ist verständlich, daß die Belegung der Pointer zuallererst bei Programmstart erfolgen muß. Dabei wird der notwendige Speicherraum gelöscht.
Der Stringraum beginnt am oberen Ende des Speicherbereiches. Hier kann auf unterschiedliche Art ein freier RAM-Bereich geschaffen werden. Außerdem ist die Größe des Stringraumes definierbar. Beides kann z. B. mit CLEAR A,B so erfolgen, wie es in Bild 4 gezeigt ist. An das Ende des Stringraumes schließt sich dann der Stack für den BASIC-Interpreter an.

5. Abarbeitung der Variablen und Felder

Beim Programmablauf werden die Variablen wie folgt behandelt: Stößt der Interpreter auf eine Variable, so sucht er, vom Anfang des Variablenraumes beginnend, nach dieser Variablen, bis er sie gefunden hat. Falls sie noch nicht initiiert war – also fehlt –, wird sie am ersten freien Platz abgelegt. Dies hat zur Folge, daß die Suchzeit für jene Variablen am geringsten ist, die beim Ablauf des Programmes zu Beginn verwendet werden. Hierin liegt der Grund, daß erfahrene Programmierer solche Variablen zunächst willkürlich belegen. Sie initiieren sie damit, d. h. sie machen sie für einen späteren Zugriff, z. B. in einer vielfach durchlaufenen FOR-NEXT-Schleife, schnell zugreifbar! Ein analoger Vorgang wird auch bei den Arrays durchlaufen. Hier kommen jedoch die Adreßrechnungen für die einzelnen Elemente hinzu. Deshalb kann es nützlich sein, zeitweilig ein einzelnes Element auf eine schnell zugreifbare Variable zu übertragen.

6. Stringoperationen und garbage collection

Bei jedem Belegen einer Stringvariablen mit einem Text geschieht zunächst dasselbe wie bei den numerischen Variablen. Das Feld hinter der Stringvariablen wird dann wie folgt verwendet: Die Länge des String in Bytes wird hinter das Variablenkennfeld eingetragen. Dann folgt die 00 und danach die nächste freie Adresse im Stringraum des RAM. Dort wird der Wert (die ASCII-Kette) abwärts im Speicherraum eingetragen. Schließlich wird die erste freie Adresse des RAM für den nächstfolgenden String (beim KC 85/2 in

```
Beispiel:
10 FOR X=33 TO 100
20 PRINT X, CHR$(X)
30 NEXT

In Tabelle ohne Space.
!----------------------------------------------------------------------!
!Adresse!Pointer!Zeilennr.    !              Programmtext       !Zeilenende!
!----------------------------------------------------------------------!
! 400   !  00   !             !               Startbyte                   !
!----------------------------------------------------------------------!
!       !       !             ! FOR ! X ! = !  33  ! TO !  100   !          !
! 401   ! 0F 04 ! 0A 00  (10)! 81  ! 58! B4! 33 33! A6 ! 31 30 30 !    00    !
!----------------------------------------------------------------------!
!       !       !             ! PRINT ! X ! , ! CHR$ !  ( !  X ! ) !          !
! 40F   ! 18 04 ! 14 00  (20)!  9E   ! 58! 2C!  CC  ! 28 ! 58 ! 29 !    00    !
!----------------------------------------------------------------------!
!       !       !             !              NEXT                !          !
! 41B   ! 21 04 ! 1E 00  (30)!               82                 !    00    !
!----------------------------------------------------------------------!
! 421   ! 00 00 !             !            Programmende          !          !
!----------------------------------------------------------------------!
```

Bild 1 Ablage eines Programms im Speicher

Bild 3 Anordnung bei Arrays. In der Feldlänge sind nicht die Bytes für den Namen und die Feldlänge, aber für die Festlegungen der Dimensionen enthalten.

```
Beispiel:

DIM A1(1,2,3), A(3), C$(4)

!--------------------------------------------------------------------!
!   Name      !Länge des!Dimen-! Variablen je      !  Anzahl der Folge-Byte   !
!             ! Feldes  !sionen! Dimension         !je 4Byte, wie bei Variablen!
!--------------------------------------------------------------------!
! 31 41 (A1) !  67 00  !  03  !04 00!03 00!02 00!    2*3*4*4=96D=60H        !
! nach 9BH ----------------------------------------------------------!
! 00 41  (A) !  13 00  !  01  !       04 00       !      4*4=16D=10H        !
! nach 17H ----------------------------------------------------------!
! 80 43 (C$) !  17 00  !  01  !       05 00       !      5*4=20D=14H        !
!--------------------------------------------------------------------!
```

3C4H) gespeichert. Diese Methode bedeutet, daß auch dann, wenn A$ wiederholt – selbst mit dem gleichen String (z. B. in einer FOR-NEXT-Schleife) – belegt wird, ständig sich der nutzbare Stringraum verkleinert. Zu einem bestimmten Zeitraum wird also auf diese Weise mit den Strings der gesamte Stringraum gefüllt. Das gilt sogar dann, wenn nur eine einzige Stringvariable mit gleichbleibendem ASCII-Text benutzt wird. Es liegt damit zu diesem Zeitpunkt im Stringraum fast nur aktuell Sinnloses, kurz Müll, Abfall oder englisch garbage. Dieser Müll ist zu beseitigen, um weiterarbeiten zu können. Es sind also alle aktuellen String optimal dicht an das obere Speicherende zu verlagern. Das entsprechende Teilprogramm des Interpreters heißt garbage collection (zuweilen auch connection). Es tritt automatisch dann in Kraft, wenn das Stringraumende erreicht wird. Es wird u. a. aufgerufen bei den Funktionen FRE (X$) und CSAVE∗X$. Existieren viele aktuelle Stringvariablen, so dauert die garbage collection relativ lange. Da der Rechner während dieser Zeit wie gestört erscheint, ist die Kenntnis dieser Funktion wichtig. Die Rechenzeit folgt in etwa dem Quadrat der Anzahl der aktuellen Stringvariablen und ist nahezu unabhängig von ihrer Länge. Für den KC 85/2 gilt etwa die folgende Aufstellung für die benötigte Zeit:

Anzahl:	100	250	500	1000
Zeit in s:	1,7	11	41	165

Die Wiederholrate der garbage collection hängt davon ab, wieviel Platz im Stringraum nach der collection noch verfügbar ist und wie häufig Strings vom Programm benötigt werden.
Das Programm #GARBAGE# ermöglicht Ihnen, Versuche hierzu durchzuführen.

7. Probleme des GOTO und GOSUB

Der GOTO-Befehl zählt heute zu den besonderes verrufenen Befehlen. Das liegt an zwei Gründen:

- Er verführt den Programmierer zu „wilden" Sprüngen. Dabei entsteht der sogenannte GOTO-Salat, bei dem das Programm z. T. völlig unübersichtlich wird. Es besteht dann die Gefahr, daß auch unkontrollierte Programmläufe auftreten können. Das soll durch eine gut strukturierte Programmierung vermieden werden. Andererseits ist bekannt, daß in LOOP-Programmen (also mit begrenzter Schleifenzahl wie bei FOR-NEXT im Gegensatz zu WHILE-WEND) für nicht primitiv rekursive Programme GOTO notwendig ist.
- In BASIC benötigt der GOTO-Befehl meist relativ viel Zeit. Er wird nämlich wie folgt abgearbeitet:

Gemäß den Pointern der Zeile wird vorn im Programm beginnend nach der Zeilennummer für den Sprung gesucht, bis sie gefunden ist. Insbesondere bei umfangreichen Programmen und bei Sprüngen zu hohen Zeilennummern wird so viel Suchzeit benötigt. Dieselben Aussagen gelten ähnlich für den GOSUB-Befehl.
Deshalb sollten im Gegensatz zu den meist üblichen Methoden, bei BASIC alle wichtigen Unterprogramme, so weit wie nur möglich am Anfang des Programmes liegen.

Noch besser ist es, die FOR-NEXT-Schleife zu verwenden. Sie zählt zu den schnellsten Abläufen bei BASIC. Bei vielen Interpretern ist es auch möglich, die unendliche Schleife gemäß

```
FOR X = 1 TO 2 STEP 0
```

zu programmieren. Sie schafft dann zusätzliche Möglichkeiten.

8. BASIC-Stack

In dem BASIC-Stack werden vor allem die folgenden Funktionen ordnungsgemäß verwaltet:

- FOR-NEXT-STEP
- GOSUB
- Offene Klammer
- Zwischenergebnisse bei Rechnungen.

Deswegen wird ein dynamischer Speicherraum erforderlich, der ausreichend sein muß und gut zu verwalten ist. Unter normalen Bedingungen geschieht dies automatisch. Ein Beispiel soll jedoch mögliche Probleme andeuten:

```
10 FOR X=1 TO 20
20 FOR Y=5 TO 35
30...
40 NEXT X
```

In diesem Fall durchsucht NEXT X den Stack solange, bis FOR X gefunden ist. Dabei werden alle zuvor stehenden Werte vom Stack entfernt. FOR Y kommt also gar nicht in Aktion.

9. Optimierung von Programmen

Leider waren die bisher durchgeführten Betrachtungen notwendig, um die folgende Zusammenfassung zu verstehen. Sie wird daher nicht sehr im einzelnen begründet, denn die meisten Fakten dürften unmittelbar einsichtig sein. Andere benötigen etwas mehr Überlegung. Sie regen dann aber vielleicht auch zum Experimentieren mit dem Interpreter an. Eine Optimierung kann nach drei z. T. widersprüchlichen Zielen erfolgen:

- Geringer Speicherbedarf
- Schneller Programmablauf
- Gute Selbstdokumentation

Die für sie nützlichen Methoden sollen jetzt getrennt in Stichworten behandelt werden:

9.1. Minimierung des Speicherbedarfs

Im Programmspeicher
- Vielfachbefehle je Zeile
- Mehrfachverwendung lokaler Variablen
- Unterprogramme (GOSUB)
- REM-Texte weglassen
- Leertasten weglassen
- NEXT ohne Variable, wenn zulässig
- Konstanten und Strings in Variablen bzw. DATA-Zeilen ablegen
- In Argumenten von PRINT, TO, STEP usw. Verknüpfungen verwenden
- DEF FN verwenden
- Variablen für mehrfach verwendete Verknüpfungen
- LET weglassen

Beim Programmablauf (statisch)
- Variablen- und Stringanzahl senken
- DIM-Anweisung richtig wählen (Null-Element)
- Bei vielen (>6) Variablen Arrays verwenden.

Im Stackbereich (dynamisch)
Je nach Interpreter benötigt jede aktive Anweisung etwa

FOR-NEXT	20 Bytes
GOSUB	6Bytes
Offene Klammer	4 Bytes
Zwischenergebnis	12 Bytes

Im Stringraum
Mit jeder Verwendung einer (auch derselben) Stringvariablen wird neuer Stringraum beansprucht. So wird der Stringraum immer weiter belegt. Am Ende des Stringraumes erfolgt garbage collection.

9.2. Schneller Programmablauf

- FOR-NEXT statt GOTO oder GOSUB verwenden
- Mit GOTO und GOSUB zu möglichst niedrigen Programmzeilen springen
- GOSUB ist schneller als zwei GOTO
- DEF FN statt GOSUB
- Häufig verwendete Variablen zu Beginn initialisieren
- Variablen statt Zahlen verwenden (Konvertierungszeit)
- Konstanten mittels Variablen ablegen
- FOR-NEXT-Schleifen (vor allem innere) einfach gestalten:
z.B. keine REM verwenden, keine GOTO, Rechnungen optimieren
- REM weglassen bzw. an das Ende von Programmen
- Bei NEXT Variable weglassen (dialektabhängig)
- Rechnungen nicht wiederholen
- Arithmetik einfach gestalten, z. B.: statt: 2∗A–> A+A; A^2–> A∗A; Multiplizieren statt Dividieren
- Logik aufgliedern und „logische" Variablen verwenden
- Feldvariablen bei häufiger Nutzung auf normale übertragen
- Wertetabellen statt wiederholter Rechnungen anlegen
- LET beschleunigt in einigen Dialekten

9.3. Gut lesbare Programme

- REM zur Erklärung von Programmabläufen
- REM zur optischen Gliederung des Programms
- Tiefe der FOR-NEXT-Schleifen durch Abstand von Zeilen-Nr. zeigen
- IF an den Anfang von Zeilen
- Programm selbst gut strukturieren (DEF FN, GOSUB, DATA usw.)
- Strukturierung durch zusätzliche Leertasten
- Variablen nicht doppelt verwenden
- Am Ende in REM wichtige Hinweise vermerken, z. B. über Benennung von Variablen.

Diskettentransfer

Marius van der Meer, Berlin

Die Nichtkompatibilität von unterschiedlichen Systemen der Kleinrechentechnik stellt oftmals bei deren Integration zum Zwecke der Rationalisierung von Informationsprozessen ein Problem dar, das nur mit hohem Aufwand gelöst werden kann. Soweit sich durch geeignete Software diese Probleme bewältigen lassen, ist eine derartige Lösung im Gegensatz zur Hardware-Kopplung um so effektiver, je häufiger sie (nach-)genutzt wird.

Als ein konkretes Beispiel dafür kann die Handhabung unterschiedlicher Diskettenaufzeichnungsformate gelten. So wurde auf Basis des DOS der Firma Technology für den ZX Spectrum und des Betriebssystems SCP auf dem PC 1715 ein Diskettenkatalog-Anpassungsprogramm geschaffen, welches den wechselseitigen Lese- und Schreib-Zugriff zu Files (z. B. Textfiles) gestattet. Diese Lösung bewährt sich unter zusätzlicher Verwendung einer Textfile-Transformierung zur Übertragung von Texten zwischen dem Textverarbeitungsprogramm Tasword auf dem ZX Spectrum und TP oder Wordstar unter SCP oder CP/M auf dem PC 1715.

Da das physische Aufzeichungsformat beider Betriebssysteme identisch ist (mit 16 Sektoren zu je 256 Bytes auf einer Spur), erfordert die Anpassung nur die Schaffung zweier gleichzeitig auf der jeweiligen Diskette verfügbarer Diskettenkataloge:

- in Spur 0, Sektoren 0 bis 7, für den CATalog des DOS
- in Spur 2, Blocknummern 00 und 01, für das DIRectory des SCP.

Mit der Belegung der Spur 0 durch den CATalog stehen die Systemspuren zwangsläufig nicht mehr für das Booten des SCP zur Verfügung, diese Disketten können also nicht im Systemlaufwerk A eines PC 1715 benutzt werden, im Laufwerk B dagegen sind sämtliche Files der Diskette über ihren Eintrag im DIRectory verfügbar. Da der CATalog in der ersten Systemspur steht, ist er für den TP-Nutzer vor irrtümlicher Zerstörung durch Überschreiben geschützt. Gleichzeitig kann er auf der Basis des gültigen DIRectory über ein Service-Programm aktualisiert werden.

Umgekehrt bilden der Rest der SCP-Systemspuren sowie das DIRectory zusammen den ersten Eintrag im CATalog des DOS von Technology Research unter einem Namen, der von der Tastatur nicht generierbare und auf dem Bildschirm nicht sichtbare Zeichen enthält und daher ebenfalls nicht ohne weiteres gelöscht werden kann. Mittels einer speziellen CAT-Service-Routine unter TR-DOS wird der CATalog in ein aktualisiertes DIRectory transformiert und in den SCP-Blöcken 00 und 01 dieses Files abgelegt.

Besondere Beachtung erfordert der Umstand, daß unter TR-DOS die Files in strenger Weise in aufeinanderfolgende Sektoren (und ggf. Spuren) fortlaufender Numerierung abgelegt werden. Zum Verdichten einer Diskette mit teilweise gelöschtem Inhalt dient daher unter TR-DOS ein Systemprogramm MOVE, welches die Bereiche gelöschter Files durch Umkopieren noch gültiger nachfolgender Files überschreibt und damit dichte Packung erzeugt. Vor jedem Transfer einer Diskette vom TR-DOS zum SCP ist daher neben der Aktualisierung des DIRectory ggf. eine solche MOVE-Routine auszuführen. Dazu wird der Nutzer beim Versuch, ein aktualisiertes DIRectory auf einer nicht verdichteten Diskette zu generieren, automatisch aufgefordert. Das ist notwendig, um zu verhindern, daß unter SCP ein neues File nicht in derselben streng fortlaufenden Weise wie unter TR-DOS, sondern eventuell unter Benutzung niedrigerer, frei gewordener Blocknummern abgelegt wird. Aus demselben Grund ist unbedingt zu vermeiden, daß bei ein und derselben Sitzung unter SCP ein anderer als der letzte Fileeintrag gelöscht und dann weitere Files erzeugt werden. In derartigen Fällen sollte stattdessen die Löschung unter TR-DOS, erfolgt von einer DIRectory-Aktualisierung, erfolgen.

In den Bildern ist der kommentierte Quelltext für das ZX-Spectrum-Programm zur Erzeugung eines DIRectory aus einem CATalog gezeigt.

1

```
  10        ORG   #B000
  20        CALL  #3C06         ;DOS-ROM ein
  30        PUSH  HL            ;Rückkehradresse
  40        XOR   A
  50        LD    (#5D0C),A
  60        CALL  #2371         ;Systemsektor holen
  70        CALL  #2E1F         ;CLS
  80        CALL  #2E0E         ;Kanal#2 (Screen)
  90        LD    HL,HEAD       ;Kopfzeile
 100        RST   #18           ;ausgeben
 110        LD    HL,DRIVE      ;Laufwerk
 120        RST   #18           ;ausgeben
 130        LD    A,(#5CF6)     ;LW-Nummer
 140        ADD   A,#41         ;als ASCII
 150        RST   #10           ;ausgeben
 160        LD    A,":"         ;:
 170        RST   #10           ;ausgeben
 180        LD    HL,#5BF5      ;Diskettennamen
 190        RST   #18           ;ausgeben
 200        CALL  #3D1B         ;CRLF auf Screen
 210        LD    HL,#5BF4      ;Adresse der
 220        LD    A,(HL)        ;Anzahl gelöschter Files
 230        AND   A             ;prüfen
 240        JR    Z,MOVED       ;ob Null
 250        LD    HL,MOVPRP     ;sonst Aufforderung
 260        RST   #18           ;zum Verdichten ausgeben
 270        JP    #2323         ;und Ende
 280 MOVED  LD    A,(#5BE4)     ;Anzahl der Files
 290        CALL  #3912         ;dezimal ausgeben
 300        LD    HL,#39DD      ;" File(s)"
 310        RST   #18           ;ausgeben
 320        LD    HL,KOPF       ;Kopfzeile Katalog
 330        RST   #18           ;ausgeben
 340        LD    IX,CPMDIR-#20;0. Position des Directory
 350        CALL  #234D         ;1. CAT-Sector lesen
 360        LD    HL,#5B00      ;Zeiger auf Anfang
 370 NEXTF  CALL  SEEKFI        ;Fileeintrag suchen
 380        CALL  #3D1B         ;CRLF auf Screen
 390        PUSH  HL
 400        CALL  #3900         ;Filenamen ausgeben
 410        POP   HL
 420        PUSH  HL
 430        LD    DE,FNAME      ;Filenamen in den Puffer
 440        LD    BC,9          ;9 Bytes
 450        LDIR                ;umspeichern
 460        POP   HL
 470        PUSH  HL
 480        LD    BC,#000D      ;Zeiger auf Sectoranzahl
 490        ADD   HL,BC         ;stellen
 500        LD    A," "         ;zwei
 510        RST   #10           ;Leerzeichen
 520        LD    A," "         ;ausgeben
 530        RST   #10
 540        LD    A,(HL)        ;Sectorzahl
 550        LD    (FLEN),A      ;puffern
 560        CALL  POSPRT        ;stellenrichtig ausgeben
 570        INC   HL
 580        LD    A," "         ;3
```

2

```
 590        RST   #10           ;Leerzeichen
 600        LD    A," "         ;ausgeben
 610        RST   #10
 620        LD    A," "
 630        RST   #10
 640        LD    A,(HL)        ;Startsectornummer
 650        LD    (ST_SEC),A    ;puffern
 660        AND   #07           ;in 8er Schritten
 670        LD    C,A           ;als
 680        LD    A,(FLEN)      ;Filelänge (Blockweise)
 690        ADD   A,C           ;aufrunden
 700        LD    (FLEN),A      ;und puffern
 710        CALL  POSPRT        ;und stellenrichtig ausgeben
 720        INC   HL
 730        LD    A," "         ;Leerzeichen
 740        RST   #10           ;ausgeben
 750        LD    A,(HL)        ;Starttracknummer
 760        LD    (ST_TRK),A    ;puffern
 770        CALL  POSPRT        ;und stellenrichtig ausgeben
 780        XOR   A             ;Nullte
 790        LD    (EXTNO),A     ;Extentnummer der Files
 800 EINTR  XOR   A             ;Standardlaufwerk A
 810        LD    BC,#0020      ;Länge je DIR-Eintrag
 820        ADD   IX,BC         ;addieren und Zeiger
 830        PUSH  IX            ;kellern
 840        LD    (IX),A        ;Laufwerknummer ablegen
 850        INC   IX
 860        LD    HL,ENTRYN     ;Anzahl der Einträge
 870        INC   (HL)          ;erhöhen
 880        LD    A,(HL)        ;und prüfen
 890        CP    #81           ;ob schon > 128
 900        JP    NC,DIRFUL     ;dann ist DIRECTORY voll
 910        LD    HL,FNAME      ;sonst Filenamen
 920        LD    B,9           ;9 Bytes
 930 NAME   LD    A,(HL)        ;holen
 940        LD    (IX),A        ;und ablegen
 950        INC   HL
 960        INC   IX
 970        DJNZ  NAME
 980        LD    A," "         ;danach zwei
 990        LD    B,2           ;Leerzeichen
1000 EXT    LD    (IX),A        ;ablegen
1010        INC   IX
1020        DJNZ  EXT
1030        LD    A,(EXTNO)     ;Extentnummer holen
1040        LD    (IX),A        ;und ablegen
1050        INC   IX
1060        XOR   A             ;und zwei
1070        LD    (IX),A        ;Nullen
1080        INC   IX            ;ablegen
1090        LD    (IX),A
1100        INC   IX
1110        LD    A,(FLEN)      ;Sectoranzahl
1120        SUB   #40           ;prüfen
1130        JR    C,LASTEX      ;wenn < oder = 64
1140        JR    Z,LASTEX      ;dann ist das letzter Extent
1150        LD    (FLEN),A      ;sonst restliche Sectorzahl
1160        LD    A,#80         ;merken, 128 Records
```

3

```
1170        LD    (IX),A        ;eintragen
1180        LD    (EXTLEN),A    ;und merken
1190        INC   IX
1200        CALL  SECNO         ;Blocknumerierung
1210        LD    HL,EXTNO      ;Extent-Zähler
1220        INC   (HL)          ;erhöhen
1230        LD    A,(ST_TRK)    ;Startspur
1240        ADD   A,4           ;um 4 weiterstellen
1250        LD    (ST_TRK),A    ;und merken
1260        POP   IX            ;Extentstartadresse
1270        JR    EINTR         ;restaurieren u nächster Eintrag
1280 LASTEX ADD   A,#40         ;Sectorzahl restaurieren
1290        SLA   A             ;verdoppeln zu Recordzahl
1300        LD    (IX),A        ;und ablegen
1310        LD    (EXTLEN),A    ;sowie merken
1320        INC   IX
1330        CALL  SECNO         ;Blocknumerierung
1340        POP   IX
1350        POP   HL            ;CAT-Eintragstartadresse
1360        LD    DE,#0010      ;um Eintraglänge
1370        ADD   HL,DE         ;erhöhen
1380        JP    NEXTF         ;und nächstes CATFile bearbeiten
1390 SEEKFI CALL  #2460         ;ggf. nächsten CATSector lesen
1400        LD    A,(HL)        ;1. Byte
1410        OR    A             ;prüfen
1420        JR    Z,ALLES       ;kein File:CAT-Ende
1430        CP    1             ;File gelöscht
1440        CALL  Z,#245B       ;dann überspringen
1450        RET   NZ            ;sonst Ende
1460        JR    SEEKFI        ;oder nächstes File suchen
1470 ALLES  LD    A,#80         ;bis 128 Fileeinträge
1480        LD    BC,#0020      ;zu je 32 Byte
1490        ADD   IX,BC         ;ab aktuellem DIR-Zeiger,
1500        PUSH  HL
1510        LD    HL,ENTRYN     ;Anzahl der Einträge
1520        LD    B,(HL)        ;holen
1530        POP   HL
1540        SUB   B             ;von 128 abziehen
1550        LD    B,A           ;und b als Zähler
1560        LD    A,#E5         ;E5 als CP/M-Leerzeichen
1570 FILL   LD    C,#20         ;in 32 Bytes je Eintrag
1580 FILLN  LD    (IX),A        ;schreiben
1590        INC   IX            ;bis
1600        DEC   C             ;DIRECTORY-Rest
1610        JR    NZ,FILLN      ;gefüllt
1620        DJNZ  FILL          ;ist
1630 SAVE   LD    C,#06         ;Sectorschreiben
1640        LD    B,#10         ;für 16 Sectoren
1650        LD    DE,#0400      ;ab Spur 4,Sector 0
1660        POP   HL            ;(Rückkehradresse zerstören)
1670        LD    HL,CPMDIR     ;aus RAM
1680        JP    #3DFD         ;schreiben
1690 POSPRT LD    C,(HL)        ;auszugebende Zahl holen
1700        PUSH  HL            ;Adresse merken
1710        PUSH  BC            ;bc kellern
1720        LD    A,C           ;Zahl merken
1730        LD    B,2           ;max. 2 Leerzeichen
1740        CP    10            ;Zahl < 10 ?
```

4

```
1750        JR    C,L2          ;ja:Sprung
1760        DEC   B             ;sonst 1 Leerzeichen weniger
1770 L2     CP    100           ;Zahl < 100 ?
1780        JR    NC,AUSG       ;nein: Ausgabe
1790 L1     LD    A," "         ;sonst Leerzeichen
1800        RST   #10           ;ausgeben
1810        DJNZ  L1            ;und ggf. nochmal
1820 AUSG   POP   BC            ;bc restaurieren
1830        CALL  #2E31         ;Zahl dezimal ausgeben
1840        POP   HL            ;Adresse restaurieren
1850        RET                 ;und fertig
1860 SECNO  LD    A,(ST_TRK)    ;Startspurnummer des Files
1870        SUB   #4            ;CP/M-Blöcke ab Spur 4 zählen
1880        SLA   A             ;zwei Blöcke je Spur
1890        LD    B,0           ;Blocknummer löschen
1900        RL    B             ;höherwertigen und
1910        LD    C,A           ;geradzahligen Teil merken
1920        LD    A,(ST_SEC)    ;Startsectornummer holen
1930        CP    8             ;<8 ?
1940        JR    C,FIRST       ;ja: 1.Block dieser Spur
1950        INC   BC            ;sonst 2. Block (zu 2KByte)
1960 FIRST  LD    A,(EXTLEN)    ;Länge des Extents holen
1970        SRL   A             ;16
1980        JR    NC,F2         ;Records
1990        INC   A             ;sind
2000 F2     SRL   A             ;ein
2010        JR    NC,F3         ;Block,
2020        INC   A             ;dabei
2030 F3     SRL   A             ;wird
2040        JR    NC,F4         ;bei Schiebeübrtrag
2050        INC   A             ;gerundet
2060 F4     SRL   A
2070        JR    NC,GLATT
2080        INC   A
2090 GLATT  LD    E,8           ;max. 8 Blöcke je Eintrag
2100 NSECNO DEC   A             ;Blockanzahl erniedrigen
2110 NSECN  LD    (IX),C        ;und Blocknummer
2120        INC   IX            ;ablegen
2130        LD    (IX),B
2140        INC   IX
2150        INC   BC            ;Blocknummer erhöhen
2160        DEC   E             ;MaxBlockzähler erniedrigen
2170        RET   Z             ;und prüfen ob schon acht
2180        CP    0             ;Blockzahl auf Null prüfen
2190        JR    NZ,NSECNO     ;ggf. weitere Blocknummer
2200        LD    BC,0          ;oder restliche sind Null
2210        JR    NSECN         ;wenn nötig leere Blocknummern
2220 DIRFUL LD    HL,FULL       ;Meldung
2230        RST   #18           ;ausgeben
2240        POP   HL            ;Rückkehradressen
2250        POP   HL            ;zerstören
2260        JP    SAVE          ;und DIRECTORY abspeichern
2270 FULL   DEFB  #0D,#0D
2280        DEFM  "CP/M-DIRECTORY IS FULL !"
2290        DEFB  0
2300 MOVPRP DEFM  "reMOVE, please!"
2310        DEFB  0
2320 EXTLEN DEFB  0
```

Bilder 1–5 Programm zur Erzeugung eines DIRectory aus einem CATalog

5

```
2330 FNAME  DEFS  #09
2340 FLEN   DEFB  0
2350 ST_SEC DEFB  0
2360 ST_TRK DEFB  0
2370 EXTNO  DEFB  0
2380 ENTRYN DEFB  0
2390 HEAD   DEFM  "Beta-Disk  TO  CP/M-DIR Transfer"
2400        DEFB  #7F
2410        DEFM  " 1987 M. van der Meer Vers. 1.0"
2420        DEFB  #0D,#0D,#0
2430 KOPF   DEFB  #0D,#0D
2440        DEFM  "Filename/typ Len St-Sec/Trk    "
2450        DEFB  0
2460 DRIVE  DEFM  "Drive "
2470        DEFB  0
2480 CPMDIR DEFS  #1000
```



Leipziger Frühjahrsmesse 1987

Im 1. Teil des Messeberichtes in MP 6/87 hatten wir über Neuentwicklungen von Bauelementen und Computertechnik informiert. Wir wollen die Messeberichterstattung fortsetzen mit dem Vorstellen lokaler Netze und peripherer Geräte.

Lokale Netze

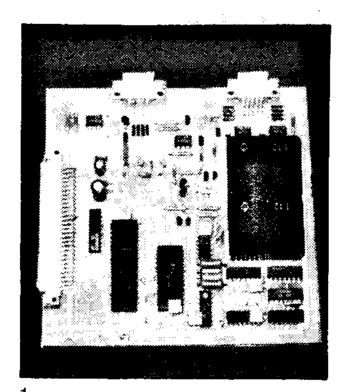
Lokale Netze gewinnen für die Datenkommunikation eine ständig wachsende Bedeutung. Sie ermöglichen den effektiven Informationsaustausch innerhalb der dezentralen Datenverarbeitung. Das Kombinat Robotron stellte mit der Anwenderlösung **ROLANET 1** (BÜRONT 1) ein solches lokales Netz vor. Für die Rationalisierung der Text- und Datenkommunikation im Büro werden mit der Lösung Dienstleistungen für folgende Aufgaben angeboten:

– Dezentrales Erfassen und Aufbereiten von Informationen
– Datei- und Nachrichtenübertragung
– Zugriff auf zentrale Ressourcen (Datenbestände, Speicher, Schnelldrucker)
– Elektronische Post mit halbautomatischer Registrierung des Posteinganges und Postausganges
– Text- und Formularbearbeitung bei zentraler Speicherung
– Informationsrecherche in zentralen Datenbeständen
– Nutzerverwaltung zur Sicherung eines wirksamen Datenschutzes.

Das ausgestellte Netz umfaßte drei PC 1715 sowie einen K 1630, der hauptsächlich als Dateiserver fungierte.
Das lokale Netz ordnet sich in das langfristig angelegte Datenfernverarbeitungskonzept der Kombinate Robotron und Nachrichtenelektronik ein. Mittelfristig wird angestrebt, über Gateway-Einheiten künftig Nutzern des Netzes einen Zugang zu
– öffentlichen Nachrichten- und Kommunikationssystemen des Kombinates Nachrichtenelektronik und
– überbetrieblichen, paketvermittelt arbeitenden Rechnernetzen zu ermöglichen.

Dabei wird zunehmend auf den Einsatz moderner Lichtwellenleiterkabel orientiert.

Das Übertragungssystem der 1. Ausbaustufe ist u. a. durch folgende Leistungsparameter gekennzeichnet:
Übertragungsrate: 500 KBd (brutto), 120 KBd (netto)
Übertragungsentfernung: max. 1000 m
anschließbare Datenendeinrichtung: robotron K 1630, A 5120, PC 1715, A 7100, 256 Teilnehmer, real 20–100
Medienzugriff: nicht determiniert, mit Kollisionserkennung, CSMA/CD-Verfahren
Datensicherung: 2 Byte CRC je Frame
realisierter Standard: nach ISO/TC 97/SC 6 DP 3802/3 (entsprechend „Ethernet").
Auf ihre spezifischen Belange zugeschnittene, aber zum Teil nachnutzungsfähige lokale Netze schufen die Akademie der Wissenschaften der DDR und mehrere Hochschulen. Bekannt sind zum Beispiel das LOTUNET der TU Dresden, das IHDnet des Informatikzentrums der TU Dresden (vormals Ingenieurhochschule Dresden) oder LANCELOT der Humboldt-Universität zu Berlin. Auf der LFM wurde nun ein weiteres, von der Ingenieurhochschule für Seefahrt Warnemünde/Wustrow entwickeltes, lokales Rechnernetz unter der Bezeichnung **SCOM-LAN** erstmalig vorgestellt. Als Zielrechentechnik dienen der PC 1715 sowie die 8-Bit-Mikrorechnersysteme auf der Basis von K-1520-OEM-Modulen einschließlich des BC A5130.
SCOM-LAN ist eine leistungsfähige „low cost"-Lösung. Es sichert bei einfachster Hardware den schnellen und zuverlässigen Datenaustausch zwischen Personalcomputern, ermöglicht dabei die Einbindung verschiedener Rechnertypen ebenso wie die kollektive Nutzung teurer Peripheriegeräte.Die auf der LFM gezeigte Beispiellösung bestand aus einer Vernetzung von fünf PC 1715 sowie einem OEM-Terminal mit 256-KByte-RAM-Disk-Simulation. Die Leistungsfähigkeit des LAN-Transportsystems wurde u. a. an den Applikationen „Intelligenter Printserver 63XX" und einem komfortablen Filetransfer demonstriert.
Die Installationsgrundlage der Vernetzung bildet das handelsübliche Sortiment der Rundfunkempfangstechnik. Das gezeigte Netzwerk hatte eine Ausdehnung von 1200 m Koaxialkabel. Einige technische Daten: 153,6 kBit/s Brutto, Datenpaketvermittlungssystem, max. 100 Stationen und CSMA/CD-Zugriff mit bzw. ohne Priorität.
Ein anwenderoffenes Transportsystem nach dem OSI-ISO-Standard (Ebene 4, Klasse 0 oder 1) bildet die Voraussetzung für betriebssystemunabhängige, anwenderspezifische Lösungen in PASCAL 880/S bzw. auf Assemblerebene.
Die Produktion der SCOM-LAN-Steckeinheit für den PC 1715 und ab dem IV. Quartal 1987 der Vertrieb erfolgen im VEB Robotron Büromaschinenwerk Sömmerda.
Die OEM-Variante wird als Nachnutzungsangebot von der Entwicklerinstitution bereitgestellt.

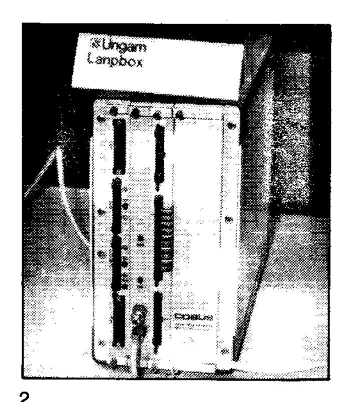


1

Bild 1 zeigt die Netzwerkleiterkarte für den PC 1715 mit zusätzlicher IFSS-Schnittstelle zur freien Benutzung.

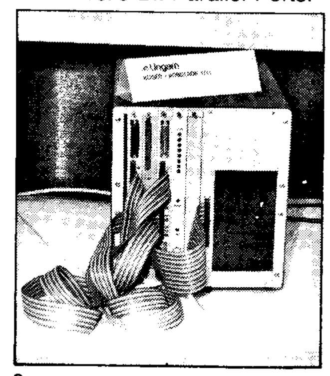
Die Bulgarische Akademie der Wissenschaften zeigte ihr für den PC Pravetz 8 M weiterentwickeltes lokales Netz **LC NET**. Der Vorteil des 8 M liegt darin, daß er zwei unterschiedliche Prozessoren (Z 80 und 6502) besitzt und damit sowohl CP/M 2.2 als auch DOS 3.3 als Betriebssysteme genutzt werden können. Besonders im Einsatz an Schulen erweist sich die Möglichkeit der Erweiterung des normalen 64-KByte-RAM um 4 mal 128 KByte zum Aufbau einer RAM-Disk als günstig. Vorteile des Netzes sind:
– gemeinsame Nutzung teurer Peripherie (Drucker, Festplatten) am Server
– Datenschutz mittels Paßworten
– Anschluß auch von PC ohne eigene Floppy-Laufwerke möglich
– Nachrichtenaustausch mittels Elektronik-Mail-Funktion.
Zum Abschluß einige technische Parameter: busorientiertes Netzwerk, Polling-Zugriffsverfahren, Teilnehmerzahl bis 64 (mittels Semaphor-Tabelle bis 16 Paßwörter möglich), Buslänge bis 3000 m, Übertragungsgeschwindigkeit bis 250 kBit/s.

Das Institut für Computer und Automatisierung der Ungarischen Akademie der Wissenschaften stellte das lokale Netz **COBUS** und Komponenten dafür vor. Auf der Messe realisierte COBUS den Datenaustausch zwischen einem PC 1715 und einem ungarischen PC namens KOSER. COBUS arbeitet im CSMA/CD-Verfahren. Die Übertragungsrate beträgt 1 MBit/s. Als maximaler Übertragungsweg sind 1000 m möglich. Die Verbindung erfolgt über normales 75-Ohm-Koaxialkabel. Maximal 99 Teilnehmer können zu einem Netzwerk zusammengeschlossen werden.



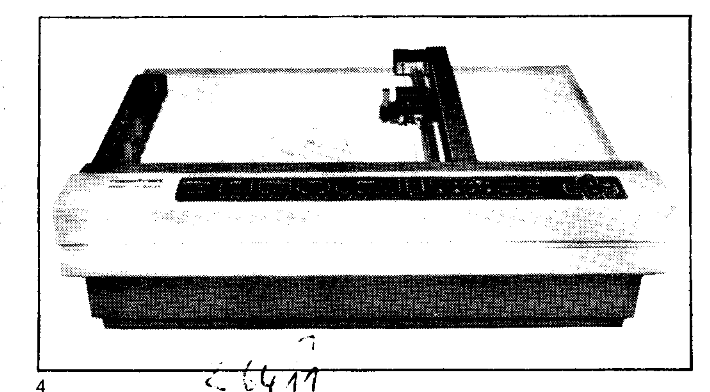
2

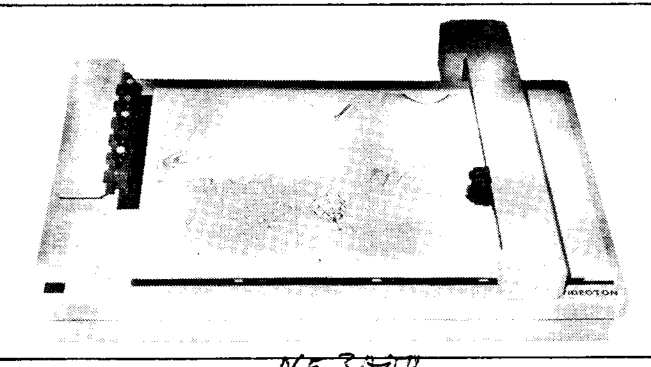
Als Kopplungseinheit dient **LANPBOX** (Bild 2) mit zwei bzw. acht V.24-Schnittstellen und ein bzw. zwei 8-Bit-Parallel-Ports.

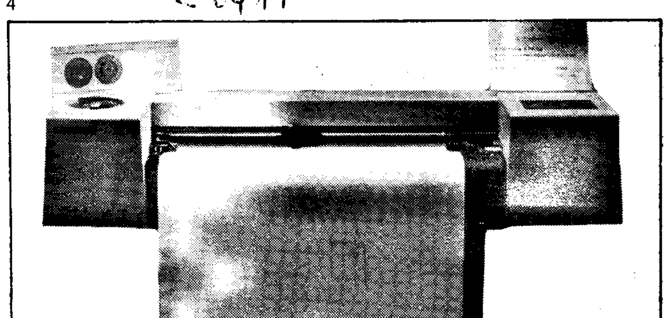


3

Außerdem war eine Disk-Server-Station (Bild 3) mit Hintergrundspeicher hoher Kapazität (Winchesterlaufwerk) zu sehen. Ma-


4 K 6411


5 NE 3000


6 K 7570 A

ximal 16 Nutzer können gleichzeitig auf den Speicher zugreifen. Die erforderliche Software ist verfügbar für CP/M und MS-DOS. Angeboten wurde auch die Leiterplatte **KONKOLY,** die notwendig ist, um IBM-PCs für den Anschluß an das lokale Netz COBUS nachzurüsten. Die Karte hat einen eigenen Prozessor – den Z 80 A, 4 MHz – und verfügt über 64 KByte RAM sowie 2 × 16 KByte ROM. Über den Prozessor wird zusammen mit einem LAN-Chip das Netzprotokoll realisiert, und über den Speicher erfolgt die Kommunikation Prozessor und PC.

Periphere Geräte

Das grafische Tablett robotron **K 6405** ist als OEM-Gerät für die interaktive Arbeit an grafischen Bildschirmplätzen konzipiert. In vier Grundbetriebsarten wird die Position des Meßwertaufnehmers auf der aktiven Fläche des Tabletts ermittelt und als digitale Information, bestehend aus einem Code und den Koordinaten, am Ausgang des grafischen Tabletts dem Anwenderprogramm bereitgestellt. In der Betriebsart POINT erfolgt die einmalige Koordinatendefinition durch Betätigen des Stift- oder Kursorschalters. RUN liefert die kontinuierliche Koordinatendefinition in Abhängigkeit von der eingestellten Abtastrate bzw. Mindestschrittweite. Die kontinuierliche Koordinatendefinition durch ständige Betätigung des Stift- oder Kursorschalters ist in TRACK realisiert. RESET ist der Grundmodus. Die Eingabe ist über Randmenü möglich, aber nicht die Datenübertragung zum übergeordneten System. Die Menüdaten des Randmenüs bestehen aus 43 alphanumerischen Zeichen, 15 Sonderfunktionen (umschaltbar) und 23 Programmfunktionen.

Einige wichtige technische Angaben sind:
Aktive Fläche: (210 × 297) mm (Format A4)
Meßprinzip: induktiv
Auflösung: 0,1 mm
Genauigkeit bei 20 °C:
bei Verwendung des Kursors ±0,5 mm
bei Verwendung des Stiftes ±0,8 mm
Genauigkeitsabweichung: 0,02 mm/K
Digitalisiervorlage: nichtmetallisch bis 0,5 mm Dicke
Abtastrate: 50 Koordinatenpaare pro Sekunde
Interface: V.24 (RS 232 C)
Datenrate: einstellbar bis 19200 Baud
Übertragungsprozedur: Asynchronbetrieb, 1 Startbit, 7 Datenbits, 1 Paritätsbit, 1 Stoppbit/Zeichen
Übertragungscode: KOI-7
Meßwertaufnehmer: Stift oder 1 Tastenkursor
Stromversorgung: extern vom übergeordneten System
Leistungsaufnahme: etwa 15 VA

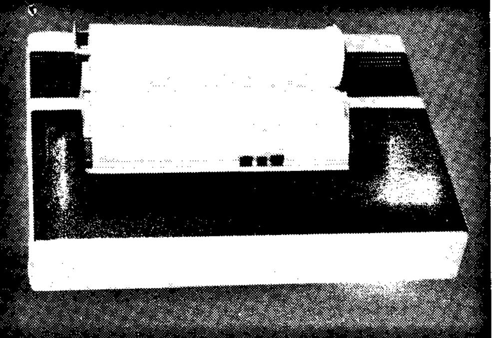
Der Plotter robotron **K 6411** (Bild 4) ist ein Auftischgerät im Format A2, das die Ausgabe grafischer Informationen in 8 Farben gestattet. Der Plotter ist mit „Intelligenz" ausgerüstet, die auf eingebauten 16- und 8-Bit-Mikroprozessoren basiert. Das Gerät verfügt über umfangreiche Standardsoftware, z. B. für Kreis- und Vektorgeneration und verschiedene Linienarten. Im Digitalisierbetrieb können mit einer Tastensteuerung X- und Y-Koordinaten beliebiger Punkte an einen Rechner übertragen werden. Es kann auf Papier, Folie oder Transparent geschrieben werden. Das Papierformat beträgt 625 × 450 mm^2, davon die ausgenutzte Schreibfläche x-Achse 594 mm und y-Achse 420 mm. Die Papierhaltung ist elektrostatisch. Als kleinste adressierbare Schrittweite ist 0,025 mm angegeben. Die Zeichengenauigkeit beträgt 0,1 Prozent, die Zeichengeschwindigkeit max. 600 mm/s in Achsrichtung. Als Interfaces werden IFSS, V.24 und IFSP genannt. Der Plotter verfügt über 11 KByte RAM und 46 KByte PROM.

Ebenfalls mikroprozessorgesteuert ist der Plotter **NE-3000** (Bild 5) von der VIDEOTON AG aus der Ungarischen VR. Die eingebauten Schnittstellen (BSI, CENTRONICS, RS-232, IEC-625) gewährleisten universelle Kopplungsmöglichkeiten. Der Plotter kann in 6 verschiedenen Farben bzw. Strichstärken zeichnen. Für das Darstellen von Kreisen, Kreisbögen und Schraffuren soll ein einziger Befehl ausreichen. Der Zeichenvorrat umfaßt auch kyrillische Buchstaben.

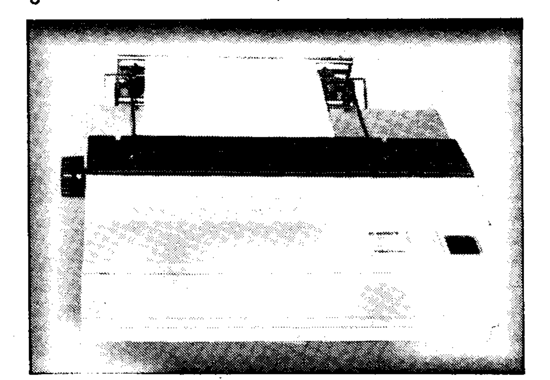
An viele Computer anschließbar ist der Plotter **7570A** (Bild 6) von Hewlett-Packard, z. B. angefangen beim ACT Apricot, über A7100, SM4 und Videoton VT32 bis hin zum Zenith **Z 150**. Das 8-Stift-Magazin ist auf der linken Seite im Gehäuse integriert. Die Genauigkeit beträgt 0,5 mm. Kleinste programmierbare Schrittgröße ist 0,025 mm. Die Zeichengeschwindigkeit ist mit 38 cm/s angegeben. In angehobener Position bewegt sich der Stift mit 51 cm/s. Ein Stiftwechsel dauert 100 ms. Die Puffergröße ist mit etwa 7 KByte beziffert. Die Papier- oder Filmbreite kann variieren von 550 bis 640 mm, die Länge von 400 bis 1000 mm.

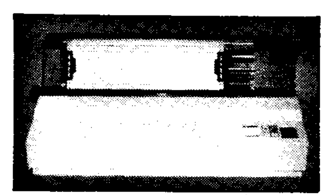
Aus dem Robotron Büromaschinenwerk Sömmerda wurde auch zur diesjährigen LFM ein breites

7


K 6304 (Thermo)

8


K6327



9 K6328

Sortiment an Druck- und Schreibtechnik präsentiert.

Der 80spaltige Thermodrucker **K 6304** (Bild 7) ist wegen seiner Fähigkeit interessant, sowohl Thermopapier als auch Normalpapier (dann mit Thermoabschmelzbandkassette) verarbeiten zu können. Im ersten Fall arbeitet er bidirektional, im zweiten unidirektional. Der sich auf der Zeile bewegende Thermospaltendruckkopf mit 10 Heizpunkten erzeugt ein Zeichenraster von 7 × 5 bei Großbuchstaben und 8 × 5 bei Kleinbuchstaben. Daneben ist der K 6304 im Bit-Image-Mode auch grafikfähig. Die Druckgeschwindigkeit beträgt 45 Zeichen/s. Als Schnittstellen sind parallel Centronics und seriell V.24 vorhanden.

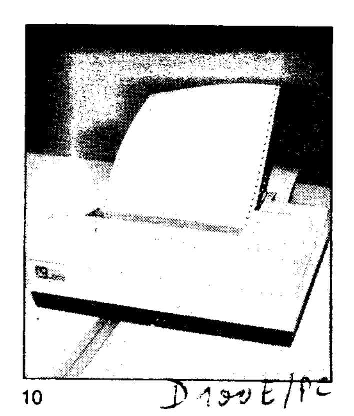
Zu der inzwischen bewährten 100 Zeichen/s schnellen Matrixdrucker-Reihe K 6310 gesellt sich nunmehr die Reihe K 6320 mit bis zu 165 Zeichen/s. Als Allzweckdrucker bieten diese Modelle Bild-, Plot- und CRT-Grafik, 6 Schriftarten, ASCII- und 8 internationale Zeichensätze sowie Papierbreiten bis zu 250 bzw. 420 mm.

Hardcopy-Drucker sind die Modelle **K 6325** und **K 6326**, die ein Schriftraster von 9 × 9 erzeugen und daneben auch grafikfähig sind.

Besonders für die Textverarbeitung geeignet sind die Drucker **K 6327** (Bild 8) und **K 6328** (Bild 9), die sich vor allem durch die verwendbare Papierbreite – 250 bzw. 420 mm – unterscheiden (80 bzw. 136 Zeichen pro Zeile). Als Schönschreibdrucker (NLQ) erzeugen sie mit 18 × 36 angeschlagenen Punkten je Zeichen ein hoch verdichtetes Schriftbild (Standard: 9 × 9 Punkte). Als Schriftartmodus kommen Normal-, Fett-, Doppeldruck, Mikroschrift (Exponenten, Indizes), Schrägschrift (Italic) und Proportionalschrift in Frage. Der Zeichenumfang ist mittels Down-Load-Funktion noch erweiterbar. Im Grafikmodus steht auch eine **8-** bzw. 9-Nadel-Ansteuerung zur Verfügung.

Der Übertragungspuffer hat 2 KByte Kapazität und kann als Option aufgerüstet werden. Schnittstellen: Centronics und V.24 (RS 232 C).

Die Formulartechnik umfaßt Friktionswalze, Traktoraufsatz, Rollenaufsatz und in Vorbereitung einen Sheet Feeder.



10 D100E/PC

Der polnische Druckerhersteller Mera Blonie stellte eine weitere Variante des zum Epson FX 80 kompatiblen Matrixdruckers D 100 vor, den **D 100 E/PC** (Bild 10). Wesentliche Neuerung ist die Erweiterung des bisherigen Zeichensatzes um den IBM- und IBM/Italic-Standard, womit der Einsatzbereich noch erweitert wird.

Im Text-Modus können 11 × 9 Punkte pro Zeichen, im Grafik-Modus 480 × 8, 960 × 8, 960 × 9 oder 1920 × 8 Punkte pro Zeile erzeugt werden. Als Schriftarten sind Normalschrift, komprimierte Schrift, Sperrschrift, komprimierte Sperrschrift, Elite und Elite-Sperrschrift vorhanden. Die Centronics-Parallelschnittstelle ist standardmäßig eingebaut, V.24 (RS 232 C) als Option möglich.

Ein weiteres Matrixdrucker-Modell wurde von dem rumänischen Außenhandelsunternehmen Electronum mit dem **IGRAF** gezeigt (Bild 11). Die normale Druckgeschwindigkeit liegt bei 150 Zeichen/s, bei Schönschreibdruck (near letter quality – NLQ) um etwa 30 Zeichen/s. In der dritten Betriebsart, dem Grafikmodus, können 60 × 72, 72 × 72 oder 99 × 72 Punkte pro Zoll gedruckt werden. **IGRAF** ist mit den Interfaces V.24 (RS 232 C) und Centronics ausgestattet.

Als Ausgabemedium für Personal- und Kleincomputer lassen sich besonders für Textverarbeitungsaufgaben günstig Schreibmaschinen verwenden, sofern sie mit einem Interface versehen sind. Vom VEB Kombinat Robotron wurden mehrere Modelle gezeigt, die dieser Forderung gerecht werden und wegen der verwendetetn Typenräder „letter quality" liefern.

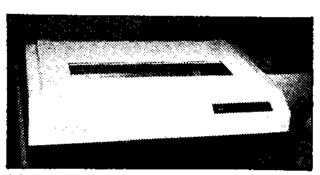
Die mit einem Linearschrittmotor als Antrieb ausgerüstete Standardschreibmaschine **S 6006** kann mit verschiedenen Moduleinschüben ergänzt werden – zum Beispiel mit einer Speichererweiterung auf 4 KByte, mit einem Fakturiermodul oder mit Interfaces wie Centronics parallel, Commodore oder V.24 (RS 232 C).

Das Spitzenmodell ist die **S 6140** (Bild 12), die gegenüber der vom Vorjahr bekannten S 6130 zusätzlich eine LCD-Anzeige erhielt, über die eine Kontrolle des Textes vor dem Ausdruck und Bedienhinweise in Klarschrift möglich werden. Die Speicherkapazität beträgt 8 KByte, davon sind etwa 7 KByte nutzbar als Arbeitsspeicher für Texte, Konstanten oder Formate.

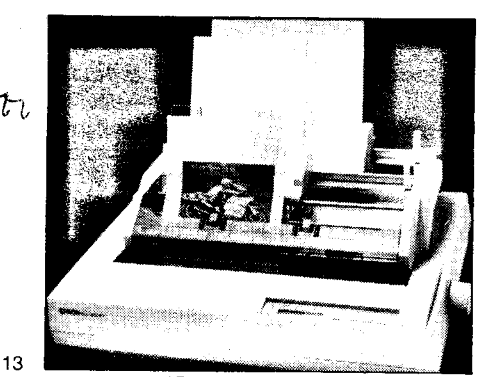
Als Option kann auch die S 6140 mit einem Interface V.24 (RS 232 C) ausgestattet werden.

Mit den Druckern **SQ-2500** (Bild 13) und **LQ-2500** (Bild 14) bewarb sich EPSON um Messegold. Der Tintenstrahldrucker SQ-2500 erreicht Druckgeschwindigkeiten von bis zu 540 Zeichen pro Sekunde. Im Schöndruck sind es bis zu 180. Der SQ-2500 formt jeden Buchstaben aus einer Matrix von 29 × 23 Punkten in einer Dichte, die 109 Punkte auf einen Quadratmillimeter bringt. Er schreibt in 5 Schriftarten, die er in einer Vielzahl von Spielarten variieren kann, z. B. in doppelter Höhe. Mit den 24 Düsen lassen sich in Bildern und Grafiken nicht nur normale und fette, sondern auch superfeine Linien ziehen.

Alle Einstellungen erfolgen über ein Tastenfeld an der Druckerfront. Auf einer 20stelligen LCD-Anzeige können die Eingaben und Betriebsarten kontrolliert werden.

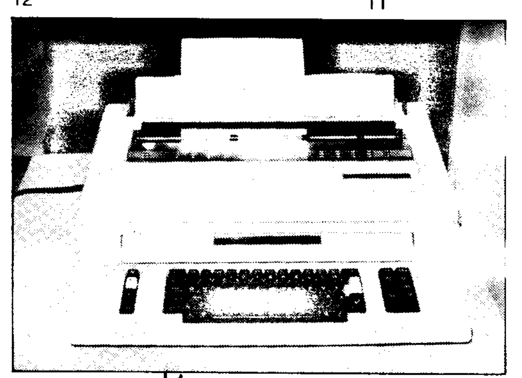


11 IGRAF

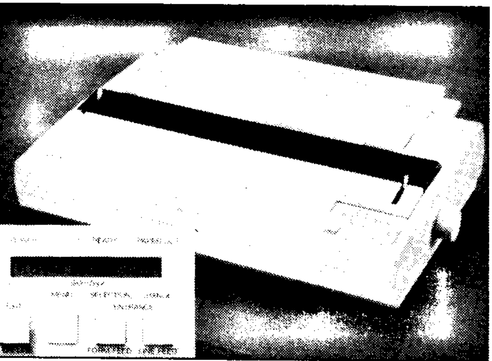


13 SQ-2500

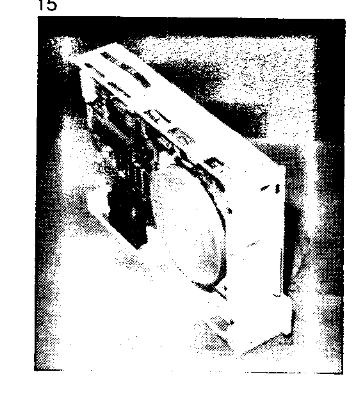
Fotos: Weiß (8), Paszkowsky (4), Werkfotos (3)



12 S6140



14 LQ 2500



15

Der LQ-2500 ist der schnellste Nadeldrucker von EPSON. Er schafft im EDV-Druck in der Schriftart Elite 324 Zeichen pro Sekunde, in Pica sind es 270. Der LQ-2500 verfügt ebenfalls über 5 Schriftarten. Das Bedienungsfeld mit integrierter LCD-Anzeige erlaubt die Wahl und die Kontrolle der Druckerfunktionen, Schrifttyp und Druckart lassen sich beim LQ-2500 mit den Tasten am Bedienerfeld einstellen, wobei die gewählte Einstellung am LC-Display abgelesen werden kann. Grafik wird genau wie Schrift bidirektional ausgedruckt.

Von **MOM** (Ungarische optische Werke Budapest) wurde mit dem **MF 8000** ein neues Mini-Floppy-Laufwerk vorgestellt – in halber Bauhöhe (slimline) und für beidseitige Aufzeichnung (Bild 15). Dank dieser konstruktiven Eigenschaften ist es gut geeignet für den Einsatz in Personalcomputern und Mikrorechnerkonfigurationen. Einige Daten: Kapazität bei FM-Codierung 500 KByte (bei MFM 1000 KByte), Datenübertragungsgeschwindigkeit 125 kBit/s (250 kBit/s), Aufzeichnungsdichte 2961 bpi (5922 bpi). Die Länge, Breite und Höhe des Gerätes betragen in mm: 210 × 146 × 41. Disketten werden entsprechend ISO/DIS 8378 verwendet. Das Laufwerk ist Shugart-kompatibel.

I. Paszkowsky, H. Weiß

CeBIT '87

Die CeBIT, die alljährlich im März in Hannover (BRD) stattfindet, überdeckt die Bereiche Datenverarbeitung, Mikro- und Personalcomputer, Telekommunikation, C-Techniken (CAD, CAM usw.), Software, Büro- und Kommunikationstechnik. Im Mittelpunkt stehen dabei mehr die zweig- bzw. anwendungsbezogenen Problemlösungen und weniger das einzelne Produkt. Nachfolgend sei auf einige Schwerpunkte der Ausstellung sowie auf Entwicklungstrends in der Hard- und Software eingegangen.

EDVA und Kleinrechner

Große EDVA wurden auf dieser Messe fast nicht ausgestellt. Von der im vorigen Jahr vorgestellten Entwicklung des Supercomputers ETA 10 von CDC war nichts zu sehen (die Firma CDC war nicht anwesend). Gezeigt wurden jedoch Computer aus dem Bereich der Minis und Superminis. Hier ist festzustellen, daß der Bereich der 32-Bit-Anlagen weiter ausgebaut und vor allen Dingen durch die VAX-Linie von DEC geprägt wird. Der Trend geht bei diesen Maschinen eindeutig zu höheren Geschwindigkeiten, um die Lücke zu den Supercomputern weiter zu verringern. Von den kürzlich in der Literatur angekündigten Minisupercomputern (sogenannte Crayettes) waren keine Modelle zu sehen. Einige der ausgestellten Systeme betrafen Einsatzfälle mit besonderen Anforderungen, vor allen Dingen bezüglich der Systemverfügbarkeit (z. B. System /88 von IBM) oder besonderer Architekturmerkmale (z. B. RISC).

Mikrocomputer und -anwendungen

Die Vielzahl von Systemlösungen und Erzeugnissen, die auf ca. 23000 Quadratmetern Ausstellungsfläche gezeigt wurden, zeugt von einem außerordentlich hohen Entwicklungstempo. Neuheiten oder auch vielfach Weiterentwicklungen bestehender Linien waren in diesem Jahr 32-Bit-Personalcomputer, Laserdrucker, lokale Netzwerke und hochauflösende Grafikanwendungen. Der Personalcomputer dringt damit in Bereiche vor, die bisher größeren Systemen vorbehalten waren und bietet immer wieder kostengünstigere und hochwertigere Lösungen.

Während im vergangenen Jahr der Personalcomputer im wesentlichen das Niveau IBM-PC und XT aufwies, ist bereits in diesem Jahr auf breiter Front eine neue Generation anzutreffen, die man als AT-286-Klasse bezeichnen könnte. Markantestes Kennzeichen ist der Intel-Mikroprozessor 80286, der sich durch eine hohe Rechengeschwindigkeit auszeichnet. So wie im vergangenen Jahr erste 80286-Systeme präsent waren, so vollzieht sich in diesem Jahr die Wende zum 32-Bit-PC durch erste Produkte mit dem Intel 80386. Dieser Hochleistungs-Personalcomputer wird einen neuen „Industriestandard" markieren, der sich vor allem durch große Haupt- und Massenspeicherkapazität sowie eine hohe Verarbeitungsgeschwindigkeit auszeichnet. Lange erwartet, aber auch in diesem Jahr noch nicht präsent, ist das Betriebssystem MS-DOS 5.0, das zu diesem zukünftigen Industriestandard gehören wird. Es wird netzwerk- und multitaskfähig sein und die großen Arbeits- und Plattenspeicherkapazitäten unterstützen.

Ein große Anzahl kleinerer Firmen bot Zusatzkarten an, die die Leistungsfähigkeit der bereits vorhandenen Personalcomputer zum Teil erheblich steigern. Dazu gehören Speichererweiterungen, Grafikkarten, Co-Prozessoren, RAM-Disks und auch eine 80386-Erweiterungskarte. Am unteren Ende der Hardwareskala sind noch vereinzelt 8-Bit-PC anzutreffen. Auf Grund des Preisverfalls ist dieser untere PC-Bereich faktisch der künftige Heimcomputerbereich, das heißt, die Grenze zwischen Heimcomputer und Personalcomputer wird verwischt. Damit ergeben sich an beiden Enden dieser Hardwareskala weitere Ausbreitungsmöglichkeiten für das Betriebssystem MS-DOS. Faktisch gibt es heute kein Softwareprodukt mehr, das nicht auf MS-DOS aufsetzt.

Die nunmehr verfügbaren PCs der 286- und 386-Klasse erschließen neue Anwendungsgebiete. Die Messe zeigte durch mehrere Exponate die Entwicklung des sogenannten Desktop Publishing. Das heißt, der Personalcomputer wird damit zur technischen Basis einer kleinen Druckerei. Dazu gehören weiter ein hochauflösender Bildschirm, ein Laserdrucker sowie die geeignete Software. Dieses Einsatzgebiet hat auch Weiterentwicklungen auf dem Peripheriesektor angeregt. Vor allem ändern sich die Anforderungen an die Bildschirme. Sie müssen wesentlich genauer und exakter als bei den bisherigen Textdarstellungen arbeiten. Derartige Anforderungen ergeben sich auch aus den unterschiedlichsten CAD-Anwendungen. Das heißt, es geht um hochauflösende Farbmonitore, die bis zu 2 Millionen Bildpunkte verzerrungsfrei auf den gesamten Schirm bringen müssen.

Auf dem Druckersektor war erkennbar, daß der Laserdrucker inzwischen auch bei „normalen" Anwendungen Fuß gefaßt hat. Preiswerteste Ausführungen kommen bereits in den Bereich um 5000 DM und dringen damit in den bislang den Typenraddruckern vorbehaltenen Bereich ein. Diese Drucker sind modular aufgebaut und können nach und nach aufgerüstet werden. Die bisher in diesem Produktsegment dominierenden Nadeldrucker sind einerseits durch weitere Qualitätsverbesserung (Trend zu 24 Nadeln) als auch andererseits durch einen Preisrutsch nach unten zu charakterisieren. Da sie außer Text auch Grafik ausdrucken können, werden sie im Zusammenwirken mit dem oben geschilderten Trend bei Laserdruckern die Typenrad- und anderen Schönschreibdrucker verdrängen. Weiterentwicklungen gab es bei Tintenstrahldruckern und Thermotransferdruckern, insbesondere, wenn es um farbliche Darstellung geht.

Mit der Erweiterung der Leistungsfähigkeit der PCs ergeben sich auch weitergehende Anforderungen an die Massenspeicher, vor allem an Disketten und Festplattenlaufwerke. Die 3,5-Zoll-Diskettenlaufwerke haben sich nicht so schnell etabliert, wie vielleicht ursprünglich angenommen; lediglich bei kleinen portablen Aktentaschencomputern werden sie vornehmlich aus Platzgründen eingesetzt. Dominant in diesem Produktsegment sind nach wie vor die 5¼-Zoll-Laufwerke. Ihre Speicherkapazität wurde erhöht und liegt bei 400 KByte bei den kleineren Personalcomputern und 1,2 MByte in der AT-Klasse. Bis auf wenige Ausnahmen dominieren die üblichen Aufzeichnungsverfahren. Erste Lösungen auf Basis von Barium-Ferrit und Senkrechtaufzeichnungen wurden angeboten. Damit lassen sich auf einer 3,5-Zoll-Diskette 4 MByte unterbringen. Bei den Festplattenlaufwerken wird kontinuierlich die Kapazität erweitert. So wurden Winchester-Laufwerke mit Kapazitäten um 50 MByte (3,5 Zoll), 100 MByte (5¼ Zoll) und 500 MByte (8 Zoll) ausgestellt. In kurzer Zeit werden im 3½-Zoll-Bereich über 100 MByte und bei 8 Zoll über 1 GByte üblich sein.

Die Personalcomputer werden immer seltener als Einzelgeräte genutzt, vielmehr sind sie auf der Basis eines breiten Spektrums von Netzwerklösungen im Nah- und Fernbereich miteinander verbunden.

Lokale Kommunikationsnetze (LAN)

Die lokalen Kommunikationsnetze zielen auf die

– Rationalisierung der Büroarbeiten
(Dabei ist der Bürobegriff nicht zu eng zu sehen, vielmehr geht es um die informationellen Prozesse in vielen Wirkungsbereichen des Menschen.)
– flexible Fertigungsautomatisierung
– dezentral eingesetzte (Personal-)Rechentechnik.

Dazu werden – und daraus erklärt sich auch die große Vielfalt von konkreten Erzeugnissen – aus verschiedenen Industriebereichen, vornehmlich aus der Nachrichtentechnik, Computertechnik sowie der Automatisierungstechnik technische Lösungen angeboten.
Das Leistungsangebot der Nachrichtentechnik resultiert aus dem Konzept der ISDN-Nebenstelle und zielt dabei vor allem auf die Bürokommunikation (Dienste zur Text- und Datenkommunikation) und die dezentrale Nutzung der Computertechnik (Datenkommunikation). Beispielsweise bietet Ericson eine LAN-Lösung für PC-Vernetzung auf der Basis ihrer ISDN-Nebenstelle an.
Das Leistungsangebot der beiden anderen Bereiche basiert auf den bereits bekannten und auch weitestgehend standardisierten 3 Grundtypen: CSMA/CD, Token Ring, Token Bus. Sie sind durch unterschiedliche Erzeugnisse von verschiedenen Firmen und einem weiten Bereich von Leistungsmerkmalen vertreten. Ein Favorit ist nicht erkennbar, vielmehr zeigt sich – wie bereits im vergangenen Jahr – eine Schwerpunktbildung:

CSMA/CD: Büroautomatisierung
Token Ring: Rechner- bzw. Geräteverbund mit Echtzeitanforderungen und größerer Last
Token Bus: Flexible Fertigung, Automatisierung (MAP).

Andere Zugriffsverfahren bzw. Topologien sind ganz selten anzutreffen (erwähnenswert wäre eventuell nur das TOP-NET von Krone, das ein Slotverfahren nutzt).
Erkennbar ist die Tendenz, sich bei der Protokollausarbeitung und Standardisierung den höheren OSI-Schichten zuzuwenden. Damit werden für den mehr problemorientierten Nutzer die unterschiedlich technischen Konzepte (OSI-Schichten 1 und 2) immer mehr verdeckt. Ein Zusammenwirken dieser unterschiedlichen Zugriffstechniken wird möglich und unter Ausnutzung verschiedener höherer Protokolle bereits demonstriert.
Als Übertragungsmedium dominiert das Koaxialkabel; Lichtwellenleitersegmente sind anzutreffen und können bei Bedarf eingesetzt werden (jedoch nicht aus Geschwindigkeitsgründen). Es gibt eine Vielzahl von Gateway-, Bridge- und Protokollkonvertierungslösungen. Eine Vereinheitlichung bzw. Bevorzugung bestimmter OSI-Schichten ist (noch) nicht erkennbar. Deutlich ausgeprägt ist jedoch eine ständige Zunahme der Prozessorleistungen bei derartigen Konvertern (bis 32-Bit-CPU!).

Telekommunikation

Im Bereich der Telekommunikation war der Schwerpunkt ISDN besonders publikumswirksam vorgestellt. Das betraf sowohl den Bereich der beweglichen Funkdienste, hier vor allen Dingen das Vorstellen der Leistungsmerkmale des im Aufbau befindlichen C-Netzes durch die entsprechenden Endgeräte, insbesondere das sogenannte Autotelefon, als auch den drahtgebundenen Bereich, hier insbesondere konzentriert auf die Leistungsmerkmale der Endgeräte. Der bereits im Vorjahr erkennbare Trend zur Konzentration von vielen Leistungsmerkmalen und Diensten im sogenannten Multifunktionsterminal hält weiter an. Technische Mittel für Bildfernsprechen sowohl für die sogenannten Videokonferenzen als auch für Bildübertragung im 64-KBit/s-Sprachkanal wurden präsentiert. Es ist jedoch einzuschätzen, daß es sich hier um Einzellösungen handelt, da bekanntlich das internationale Normungsgeschehen den Bereich der Breitbandkommunikation noch nicht erreicht hat.
Erste Lösungen für *voice mail services* wurden gezeigt. Die technischen Mittel für die Übertragung der digitalen Datenströme sind bis in die Hierarchiestufe 500 MBit/s sowohl für Koaxleitungen als auch für Glasfaser und Richtfunk vorhanden. SEL zeigte einen Laboraufbau für eine Hochgeschwindigkeitsübertragung über einen 70 km langen Lichtwellenleiter mit einer Geschwindigkeit von 5,2 GBit/s.

C-Techniken

Unter diesem Begriff werden zunehmend alle Teilbereiche des computerunterstützten Konstruierens, Planens, Produzierens usw. zusammengefaßt. Über 170 Aussteller zeigten zu diesem Komplex ihre Erzeugnisse und Systemlösungen. Entwicklungen bei peripheren Geräten sind in starkem Maße an den aus diesen Gebieten resultierenden Anforderungen orientiert. Es ist schwierig, den rationalen Kern und den Stand der Entwicklung und des Einsatzes derartiger Lösungen aus den werbetechnisch hervorragend gestalteten Präsentationen abzuheben. Klar scheint zu sein, daß man von einer durchgängigen CIM-Lösung noch weit entfernt ist. Das ist sowohl der noch nicht abgeschlossenen Standardisierung der Kommunikationsprotokolle als auch der noch nicht durchgängig erarbeiteten Analyse der zu rationalisierenden informationellen Prozesse geschuldet. Der Weg für die Standardisierung der Kommunikationsdienste ist durch MAP und TOP gewiesen; an den endgültig zu verabschiedenden Protokollen wird jedoch noch intensiv gearbeitet.

Zusammenfassende Wertung

Folgende allgemein formulierte Tendenzen sollen eine Gesamteinschätzung geben:

- Computer- und Kommunikationstechnik entwickeln sich eindeutig aufeinander zu. In der Computertechnik prägen sich die Hauptgebiete Personalcomputer (16 bzw. 32-Bit-Verarbeitungsbreite, einige MIPS) und Minicomputer (entsprechend VAX-Spektrum) weiter aus.
- Bei Mikroprozessoren und ihrem Einsatz in PC ist festzustellen, daß eine leistungsgesteigerte 16-Bit-CPU Standard, eine 32-Bit-CPU vorhanden und eine 8-Bit-CPU in Personalcomputern faktisch nicht mehr anzutreffen ist. Es werden in beträchtlichem Umfang Schaltkreise von Intel, teilweise auch von Motorola und nahezu keine von Zilog eingesetzt. Die Zunahme der Leistungsfähigkeit von PC und peripheren Geräten erschließt neue Einsatzgebiete und stimuliert damit den Softwaremarkt beträchtlich. Anwendungssysteme existieren für nahezu alle Bereiche der Industrie, des Bauwesens, Handels, Verkehrs und des Dienstleistungsbereiches,
- Bei den lokalen Netzen geht der Trend zur Ausgestaltung der oberen Schichten des OSI-Modells und damit zu einer Vereinheitlichung der Einsatzbedingungen für die vielen technischen Realisierungen.
- Die Drahtnachrichtentechnik wird vor allen Dingen durch das ISDN-Konzept geprägt. Vorlaufarbeiten in diesem Bereich betreffen Probleme der Breitbandkommunikation. Für diese Technik scheint sich als Basiskanal 140 MBit/s herauszukristallisieren.
- Im Bereich der C-Techniken wird intensiv, vor allem im Softwarebereich, gearbeitet. Es dominieren eindeutig Anwendersysteme für einzelne Teilbereiche. Bis zur durchgängigen Lösung ist sicher noch ein langer Weg zu gehen.

Prof. Dr. P. Neubert

Journal of New Generation Computer Systems

Ab Januar 1988 erscheint im Akademie-Verlag Berlin die neue internationale Fachzeitschrift JOURNAL OF NEW GENERATION COMPUTER SYSTEMS, herausgegeben im Auftrage des Koordinierungsrates für Rechentechnik und Informatik der Akademie der Wissenschaften der sozialistischen Länder am Zentralinstitut für Kybernetik und Informationsprozesse der Akademie der Wissenschaften der DDR.
JOURNAL OF NEW GENERATION COMPUTER SYSTEMS (JNGCS) dient der Veröffentlichung von Beiträgen, die der Entwicklung von Rechnersystemen mit qualitativ neuen Eigenschaften und neuartigen Anwendungsgebieten, d. h. der Schaffung neuer Generationen von Rechnersystemen gewidmet sind.
JNGCS berichtet über Fortschritte, Erfahrungen und neueste Ergebnisse bei der Realisierung nationaler und internationaler Forschungs- und Entwicklungsprogramme sowie von Projekten, speziell von Gemeinschaftsprojekten der sozialistischen Länder, deren Ziel die Schaffung innovativer Rechnersysteme ist. Das Hauptziel der Zeitschrift ist der Informationsaustausch zu neusten Entwicklungen der Hard- und Software sowie zu Anwendungen auf dem Gebiet der Verarbeitung von Wissen und großen Datenmengen.
Daraus ergibt sich für die Zeitschrift folgendes Profil:

– Höchstintegrierte Schaltkreise (VLSI)
– Neue Entwurfs- und Fertigungstechnologien für Rechnersysteme
– Innovative Architekturen für verschiedene Anwendungen
– Softwaretechnik mit Künstlicher Intelligenz
– Wissensverarbeitung
– Mensch-Maschine-Kommunikation
– Theorie und Künstliche-Intelligenz-Forschung.

Die Zeitschrift enthält ferner Konferenzberichte, Veranstaltungskalender, Buchrezensionen u. a. Beiträge, die Forschungsergebnisse oder neue Technologien vorstellen. Übersichtsarbeiten, Berichte und Mitteilungen sind in Russisch oder Englisch zu richten an: Dr. W. Kolbe, Redaktion „JNGCS" ZKI AdW; Kurstraße 33, PF 1298, Berlin, 1086, DDR.

Praxis der Softwareentwicklung

von G. Rothhardt, VEB Verlag Technik Berlin 1987, 1. Auflage, 296 S., 30,– M

Mit diesem Titel liegt ein Fachbuch vor, das vom Praktiker für Praktiker geschrieben ist. Das geschieht in einer Art und Weise, die dieser Tatsache adäquat ist: Der Stoff ist mit methodischem Geschick unter Benutzung von authentischen und konstruierten Beispielen so aufbereitet, daß man eigene Erfahrungen bestätigt erhält und Lösungen bzw. Lösungsvorschläge angeboten bekommt. Viel Originalität beweist der Autor mit seinen bildhaften Darstellungen, die Anschaulichkeit und Einprägsamkeit hervorragend unterstützen. Nicht zuletzt durch den sprachlichen Stil, einschließlich der Ansprache des Lesers, wird die Lektüre auch demjenigen Praktiker, der einem umfangreicheren Fachbuch eventuell skeptisch gegenübersteht, leicht gemacht. Nach einem Abschnitt mit einführendem Charakter, in dem die Strukturträger des Fachgebietes *Softwaretechnik* abgesteckt werden, erfolgt dann die weitere Behandlung und Orientierung an Methoden, Hilfsmitteln und Werkzeugen des Softwareentwicklungsprozesses.
Abschnitt 2 stellt auf der Basis eines sehr weit gefaßten Methodenbegriffs bekannte und international akzeptierte Ergebnisse der Softwaretechnik vor. Moderne Aspekte dabei sind durchweg berücksichtigt. So werden sehr ausführlich Fragen der Dialogarbeitsweise von Programmen unter der Überschrift *Softwareergonomie* behandelt, und auch die Qualitätssicherung erhält den ihr gebührenden Platz. Die Behandlung von Notwendigkeit und Möglichkeiten der Nachnutzung von Software verdient ebenfalls hervorgehoben zu werden. Zu den Werkzeugen des Softwareentwicklungsprozesses wird ein systematisierender Überblick gegeben.
Abschnitt 3 wendet sich speziell an die Leiter im Softwareentwicklungsprozeß, wobei diesen auch Teile des vorangegangenen Abschnittes zu empfehlen sind. Wenn auch der eine oder andere Vorschlag kritisch gesehen werden kann (z. B. 1 bis 2 künftige Nutzer in die Entwicklung einbeziehen), so wird doch überzeugend klar: Softwareentwicklung ist mehr, als ein Programm zu schreiben! Der Abschnitt *Inbetriebnahme des Softwareproduktes* öffnet dem Letzten dabei die Augen.
Der Abschnitt 4 will Hilfestellung geben bei der Durchsetzung bzw. Einführung von neuen Methoden. Zur Erreichung dieses Zieles ist er sehr weit gefaßt, so daß nicht alles, was dort steht, spezifisch für die Softwaretechnik gilt (z. B. Qualifizierung, Motivieren). Sehr gut und wertvoll sind die Ausführungen unter der Überschrift *Vereinheitlichen* (Standards!). Gleiches gilt für die Checklisten im Anhang 2.
Insgesamt liegt mit dem Titel ein sehr gutes Informatik-Fachbuch vor, das den in der Softwareentwicklung Tätigen Bestätigung und inhaltliche Orientierung sein wird.

Dr. R. Trautloff

Die Programmiersprache FORTH

von R. Zech, Franzis-Verlag München 1985, 2. neu bearbeitete und erweiterte Auflage, 336 S.

Dieser Titel enthält in 7 Abschnitten eine umfassende Darstellung von Aufbau, Implementierungsgrundlagen, Programmierkonzepten und Eigenschaften eines FORTH-Systems. Im Vordergrund stehen eine Beschreibung der wichtigsten Verarbeitungsoperationen sowie das vollständige Glossar des FIG-FORTH-Systems. Die Wahl des FIG-Standards kann unterstützt werden: Nach wie vor ist er am weitesten verbreitet. Dem allmählichen Übergang zu FORTH-83 trägt ein Abschnitt Rechnung, in dem die wichtigsten Modifikationen und Erweiterungen anderer FORTH-Systeme (darunter auch FORTH-79 und FORTH-83) zusammengestellt sind. Weitere Schwerpunkte sind die Problemkreise Programmstrukturen und strukturierte Programmierung, Compilationsprinzipien (einschließlich einer detaillierten Erläuterung der Funktion und der Handhabung der Definitionswörter), Zahleneingabe- und -ausgabekonvertierungen sowie I/O-Konzepte (Massenspeicherverwaltung). Im Unterschied zu Brodies *Programmieren in FORTH* (Hanser-Verlag 1984) ist das methodische Konzept hier darauf gerichtet, FORTH an sich selbst zu erklären: Für viele, vor allem für komplizierte Funktionen ist der FORTH-Quelltext angegeben. Die Erklärung des Innenlebens der Wörter ist ein wichtiger Aspekt der Systemdarstellung. Übungsaufgaben mit Lösungen sowie einige Beispielprogramme (Taskverwaltung, Stringpaket, FORTH-Assembler) geben Anregungen für die eigene Programmierpraxis. Für größere Softwareprojekte (vor allem im professionellen Bereich) sind jedoch Informationen zu weiteren Fragen und Problemen notwendig, die in diesem Titel nicht dargestellt sind: Software- und Implementierungsstrategie, Faktorisierungs- und Modularisierungsprinzipien, Quelltextgestaltung und Namenskonventionen.

G.-U. Vack

Bussysteme in Mikrorechner-Automatisierungsanlagen

von J. Sawatzky, Heft 16 der Schriftenreihe der Betriebssektion der KDT des VEB GRW „Wilhelm Pieck" Teltow 1986, 86 S., Preis: 8,– M

Der im Jahre 1986 mit der Barkhausen-Medaille für die Verteidigung einer der anspruchsvollsten Dissertationen auf technischem Gebiet in der DDR geehrte Autor legt nunmehr mit dem Heft 16 eine längst fällige Veröffentlichung zur Problematik der busgebundenen Datenübertragung vor.
Dezentral strukturierte mikrorechnergestützte Automatisierungsanlagen basieren auf der digitalen Informationsverarbeitung, -übertragung und -speicherung. Die digital-serielle Informationsübertragung wird in derartigen Strukturen durch Bussysteme realisiert. In der Broschüre geht der Autor ein auf

- Stellung und Bedeutung der seriellen Datenübertragung in Automatisierungsanlagen
- Strukturen und Funktionen der Bussysteme
- das Schichtenmodell der Datenübertragung
- Standardisierungen von Datenübertragungssystemen
- ausgewählte Anlagenbussysteme sowie
- Entwicklungstendenzen bei Anlagenbussystemen.

Ausgehend von dem im Prozeßleitsystem „audatec" des VEB GRW „Wilhelm Pieck" Teltow verwendeten audatec-Bussystems, wurden die Parameter leistungsfähiger Anlagenbussysteme ausgewählter internationaler Prozeßleitsysteme miteinander verglichen. Auf Notwendigkeit, Stand und Perspektive der Standardisierung von digital-seriellen Datenübertragungssystemen wird eingegangen. Der Autor schätzt ein, daß neben den weit verbreiteten Anlagenbussystemen die Feldbussysteme zunehmend an Bedeutung gewinnen werden. Abgerundet wird die Broschüre durch ein Abkürzungs-, Bild- und Tafelverzeichnis sowie ein umfangreiches Literaturverzeichnis, das dem Autor ein tiefgründiges Quellenstudium bescheinigt.
Ein Sachwortverzeichnis, das dem Leser die Möglichkeit des kurzfristigen Einstiegs in die anspruchsvolle Thematik erlaubt, wird vermißt. Die Problematik der digital-seriellen Datenübertragung wird so verständlich und übersichtlich dargestellt, daß sie einen breiten, technisch allgemein interessierten Leserkreis anspricht. Zu beziehen ist die Veröffentlichung über die Betriebssektion der KDT oder die Zentrale Informationsstelle Wissenschaft und Technik des VEB GRW „Wilhelm Pieck" Teltow, Oderstraße 74–76, Teltow, 1530.

L. Blackert

Computerliteratur aus der VR Bulgarien

Der Staatsverlag „Technika" ist der größte Verlag für technische Literatur in der VR Bulgarien. Unter seinem Verlagssignet sind im Verlauf von über 25 Jahren mehr als 15000 Titel verlegt worden. Im Verlagsplan sind Bücher verschiedener Editionsrichtungen vertreten – wissenschaftliche, populärwissenschaftliche und Handbücher, Lehrbücher für Universitäten und Hochschulen sowie Schulbücher. Aus der Reihe *Mikrocomputertechnik für alle* wurden bisher u. a. die folgenden Titel herausgegeben bzw. sind in Vorbereitung: Die Handhabung von Personalcomputern; Das Programmieren – einfach und gleichzeitig kompliziert; Die Mikroprozessoren – das Kernstück des Personalcomputers. Eine Reihe von Veröffentlichungen erschienen bzw. werden vorbereitet zum bulgarischen PC Prawez-82. Hier können genannt werden: Wie funktioniert Prawez-82?; Diskettenoperationssystem; 50 Programme für Personalcomputer; PASCAL für Personalcomputer; Der Computer spielt, zeichnet und macht Musik; Peripheriegeräte für Personalcomputer. Natürlich fehlt auch ein BASIC-Buch nicht im Angebot. Als Beilage zum Titel BASIC für Mikrocomputer wird eine Diskette mit den textillustrierenden Programmen angeboten.

MP

ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 6
S. 161–192

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP



Heft 7 · 1987

Mikroprozessortechnik

VEB Verlag Technik Berlin

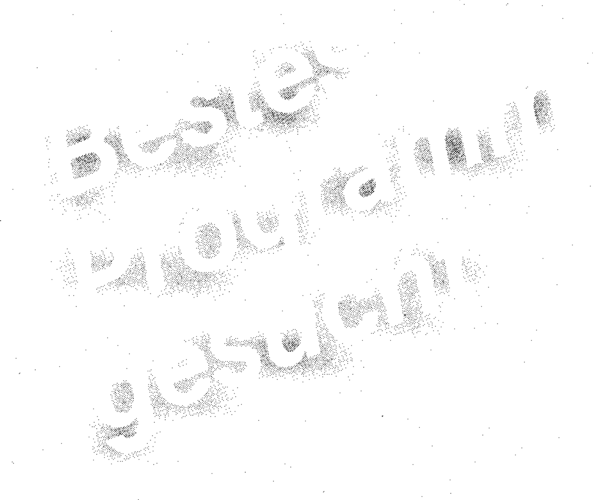
ISSN 0233-2892

- **Integrierte Schaltungen**
  Stand und Ausblick
- **16-Bit-Mikrorechner**
  Hard- und Software
- **Wechselplattencontroller für 8- und 16-Bit-Mikrorechner**

Zum Abschluß des MP-Kurses ***Programmierung in C*** möchten wir Ihnen zwei kleine Programmieraufgaben vorschlagen, bei denen Sie Ihr erworbenes Wissen unter Beweis stellen können. Senden Sie Ihre Lösungen der Aufgaben 1 und 2 bitte bis zum 25. September 1987 unter dem Kennwort ***C-Programmierung*** an die Redaktion.

Hier noch einmal unsere Anschrift:

VEB Verlag Technik
Redaktion MP
Oranienburger Straße 13/14
Berlin
1020



Unter den richtigen Einsendern werden unter Ausschluß des Rechtsweges 10 Bücher des Titels ***UNIX und C – Ein Anwenderhandbuch*** von L. Claßen und U. Oefler, 1987 erschienen im Verlag Technik, ausgelost. Die Namen der Gewinner und ausgewählte Programmlösungen werden in einer Ausgabe der MP veröffentlicht. Beachten Sie bitte, daß die bei uns eingereichten Programme frei von Rechten Dritter sein müssen.
Bei den zugesandten Lösungen setzen wir Ihr Einverständnis für deren eventuelle Veröffentlichung voraus. Testen Sie nach Möglichkeit Ihre Programme vorher am Computer (nicht Bedingung!). Senden Sie uns in diesem Fall Angaben über Computer- und Compilertyp und ein Abarbeitungsprotokoll – mit selbstgewählten Daten – zu.
Bitte vergessen Sie nicht, auf den Lösungen Ihren Namen, Anschrift, Alter und Beruf/Tätigkeit anzugeben.
Für gegebenenfalls erforderliche Rückfragen ist die Angabe einer Telefonnummer wünschenswert.

Viel Erfolg bei der Lösung der beiden Aufgaben wünscht Ihnen
Ihre Redaktion MP

Aufgabe 1
Suchen von Sachwörtern

Das Programm 12.4 soll so ergänzt werden, daß bei der Ausgabe des Textfiles nach Sachwörtern gesucht wird, die in einer Tabelle

static char *swtab [] = {"Dimension", "Feld", ...};

zur Vereinfachung der Aufgabe definiert sein sollen.
Jedes gefundene Sachwort soll mit Angabe der Seitennummer auf ***stdout*** protokolliert werden. Ein Sachwort gilt als gefunden, wenn es in den spezifizierten Zeichen identisch ist, z. B. tritt in dem Text das Wort ***„Felder“*** auf, so ist das Sachwort ***„Feld“*** enthalten und es gilt als gefunden.

Aufgabe 2
Versuchsauswertung

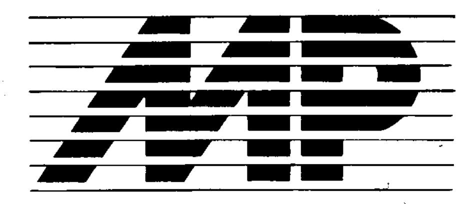
In einem File ***TEST.DAT*** stehen die Meßergebnisse eines physikalischen Versuchs im ASCII-Kode. Jeder Datensatz enthält zwei Meßergebnisse in ***float***-Darstellung. Die Anzahl der Meßergebnisse muß aus der Anzahl der eingelesenen Datensätze ermittelt werden. Die maximale Anzahl der Meßergebnisse ist kleiner als 2000.

Es sind folgende Auswertungen durchzuführen:

- Ermittlung des minimalen und maximalen Wertes,
- Berechnung des Mittelwertes,
- Bestimmung der mittleren quadratischen Abweichung vom Mittelwert.

Die Meßergebnisse sollen in 5 Spalten (Spaltenbreite 12 Zeichen, 3 Zeichen nach dem Dezimalpunkt) zusammen mit den Ergebnissen der Auswertung gedruckt werden.

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0233-2892



Mikroprozessortechnik, Heft 7 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR-1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 2 87 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)

Beirat Dr. Ludwig Claßen, Prof. Dr. sc. Dietrich Eckhardt, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Jürgen Zaremba

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

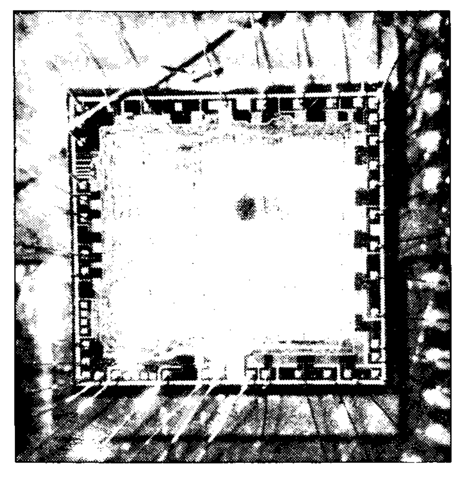
Redaktionsschluß 18. Mai 1987

AN (EDV) 49837

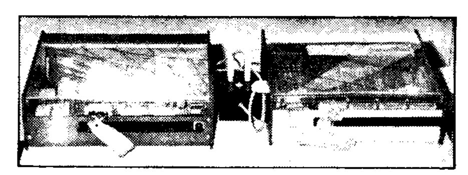
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

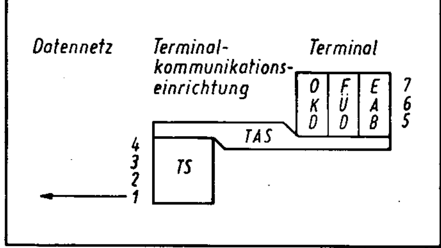
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandititi te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ústřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hà Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141-167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR-7010, und Leipzig Book Service, Talstraße 29, DDR-7010 Leipzig



Seite 195



Seite 200



Seite 219

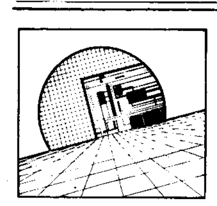
Vorschau

In MP 8/1987 haben wir für Sie einen Schwerpunkt über Einchipmikrorechner vorbereitet; u. a. finden Sie folgende Beiträge zu dieser Thematik:

– Programmentwicklung und Test für EMR U8840
– ROM-Schaltkreis U 2365 D 45 BM 200 für EMR UB 8830

Außerdem beginnen wir mit einer neuen Reihe zu REDABAS unter der Rubrik MP-Kurs.

Inhalt



12. Mikroelektronik-Bauelementesymposium

In dem Plenarvortrag zur Eröffnung des unter seiner Schirmherrschaft stehenden 12. Mikroelektronik-Bauelementesymposiums, das vom 11. bis 13. Mai im Sport- und Ausstellungszentrum Frankfurt (Oder) stattfand, betonte der Minister für Elektrotechnik und Elektronik, Felix Meier: „Die Aufgaben des Industriebereiches Elektrotechnik und Elektronik sind vor allem die beschleunigte Entwicklung, Produktion und Anwendung der Mikroelektronik mit dem Schwerpunkt der Sicherung der volkswirtschaftlichen Programme, der Versorgung der Bevölkerung mit hochwertigen Konsumgütern und des Exports. Dieser Prozeß wird wesentlich davon getragen, daß Mikroelektronik, moderne Rechentechnik und rechnergestützte Konstruktion, Projektierung und Steuerung der Produktion mehr und mehr das Leistungsvermögen der Volkswirtschaft bestimmen."

Dem gemeinsam vom VEB Kombinat Mikroelektronik und dem Bezirksvorstand der Kammer der Technik Frankfurt (Oder) durchgeführten Symposium schloß sich am 14. und 15. Mai eine Fachtagung zum gleichen Thema an. Teilnehmer dieser größten applikativen Veranstaltung der DDR mit 20jähriger Tradition waren 2400 Forscher und Praktiker aus nahezu allen Bereichen der Volkswirtschaft, der Akademie der Wissenschaften der DDR sowie aus Universitäten und Hochschulen. In 46 Vorträgen sowie zahlreichen Diskussionen und Fachgesprächen wurden neueste wissenschaftliche Erkenntnisse auf dem Gebiet der Entwicklung, Produktion und Anwendung der Mikroelektronik vermittelt, Erfahrungen beim Einsatz mikroelektronischer Schaltkreise ausgetauscht. Eine Ausstellung informierte über das zur Verfügung stehende Sortiment aktiver elektronischer Bauelemente und zeigte in der Praxis erprobte Anwendungsbeispiele des volkswirtschaftlich breiten Mikroelektronik-Einsatzes.

Weitere Informationen, insbesondere über die Ausstellung, werden in MP 8/87 veröffentlicht.

MP

Programmierolympiade und KC-Hardware-Wettbewerb

Im Rahmen der Tage des Bezirkes Dresden anläßlich der 750-Jahr-Feier-Berlins führen der Kulturpalast Dresden und das Urania-Vortragszentrum Dresden am 1. und 2. August 1987 in der Kongreßhalle am Alexanderplatz die Veranstaltung „simultan-wissenschaft-live" durch. Während der Veranstaltung werden unter Federführung der Technischen Universität Dresden zwei Leistungsvergleiche unter dem Motto „Wir beherrschen den Computer" für Software- und Hardware-Interessenten stattfinden. Das Informatik-Zentrum des Hochschulwesens an der Technischen Universität suchte in einer Programmierolympiade den besten Programmierer aus Studium, Lehre und Praxis, aus den Computerclubs und dem Kreis der Hobbyanwender, die am 1. und 2. August ihr Können an Kleincomputern KC 87 des VEB Kombinat Robotron demonstrieren werden. Von der Sektion Informationstechnik der TU Dresden wurde der beste Spezialist für einen Kleincomputer-Hardware-Wettbewerb gesucht. Gefragt waren selbstentworfene und gebaute Hardwarelösungen und Ergänzungen zum KC.

Anmerkung: Leider ging der Redaktion die Mitteilung über die Wettbewerbe verspätet zu, so daß wir einen Aufruf zum entsprechenden Zeitpunkt nicht veröffentlichen konnten.



Büchertreff am Fernsehturm

Anläßlich des 750jährigen Bestehens von Berlin finden vom 21. Juli bis zum 4. August 1987 im Ausstellungszentrum am Fernsehturm folgende Veranstaltungen statt:

- Ausstellungsprogramm mit *Verlage stellen sich vor, Berlin im Buch* und *Schönste Bücher der DDR*
- Rahmenprogramm mit Autorenlesung, Signierstunden, Buchpremieren und literarisch-musikalischen Veranstaltungen
- Buchverkauf
- Polygrafische Werkstatt in Aktion.

Der *Büchertreff* ist täglich von 10 Uhr bis 19 Uhr geöffnet.

MP

Japan beschleunigt Entwicklung von PC

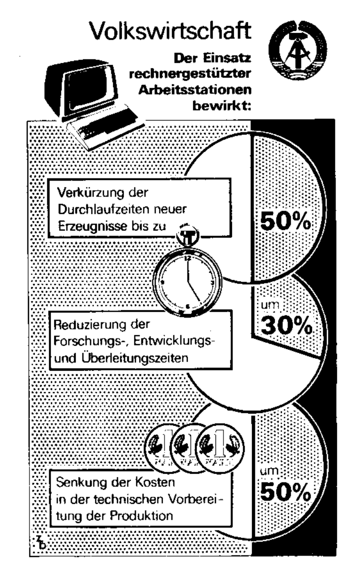
Personalcomputer sind für Elektronikkonzerne in den USA und Japan ein lukratives Geschäft geworden. Seit ihrer Entwicklung vor zehn Jahren sind in Japan über fünf Millionen PC zum Einsatz gekommen, allein 1986 wurden 1,5 Millionen abgesetzt. Ein Großteil davon waren schon Modelle der 16-Bit-Generation.

Als Anfang April in den USA die International Business Machines (IBM), die von Anfang an wesentlich an der Entwicklung der PC-Technik beteiligt ist, eine neue Generation dieser Computer – nun 32-Bit-Typen – vorstellte, urteilten japanische Zeitungen übereinstimmend: die USA blasen zum Sturm gegen die japanische Konkurrenz. Mit der neuen PC-Generation will IBM den Markt für diese Rechner wieder zurückerobern. IBM hatte selbst in den USA in den vergangenen Jahren vor allem gegen japanische Unternehmen an Boden verloren, die die 16-Bit-Computer technisch attraktiver, leistungsstärker und vor allem billiger anboten. Die Standards der neuen 32-Bit-Technik wurden von IBM strengstens geheimgehalten, um zu verhindern, daß sich die Konkurrenz schon vorab dem IBM-Standard anschließen kann.

Die japanischen Konzerne hatten schon vor Monaten angekündigt, beim Lauf um die leistungsstärkeren PC nicht zurückbleiben zu wollen. Ende des Jahres, so ließ der Verband der japanischen Elektronikindustrie mitteilen, werden die großen japanischen Computerproduzenten eigene 32-Bit-Entwicklungen auf den Markt bringen. Vor einigen Wochen meldete der japanische NEC-Konzern, einer der größten Chip-Produzenten, den Abschluß der Entwicklung eines 32-Bit-Schaltkreissystems, das mit 6,6 Millionen Operationen je Sekunde zu den schnellsten der Welt gehört. Der Übergang zu den 32-Bit-Computern – bedingt durch den scharfen Konkurrenzkampf mit USA-Herstellern – erfolgt überhastet, räumten japanische Spezialisten ein. Die Softwareentwicklung kommt mit den schnellen Fortschritten im Chip-Bau nicht nach. Ein größeres Programmangebot für die 32-Bit-Technik, mit denen die Leistungskraft erst voll genutzt werden kann, wird erst zu Beginn der 90er Jahre bereitstehen.

Was die neuen PC von IBM betrifft, so rechnen sich japanische Hersteller dennoch Chancen aus. Grund: Die Diskettenlaufwerke in den IBM-Maschinen sind teilweise „Made in Japan".

ADN



Einsatz rechnergestützter Arbeitsstationen

Schlüsseltechnologien wie Mikroelektronik, moderne Rechentechnik und rechnergestützte Konstruktion, Projektierung und Steuerung der Produktion bestimmen immer stärker das Leistungsvermögen der Volkswirtschaft der DDR. Die Grafik zeigt, was mit dem Einsatz rechnergestützter Arbeitsstationen u. a. bewirkt werden soll.

Grafik: ADN-ZB

Datenübertragung

Die Sektion Physik der Martin-Luther-Universität Halle-Wittenberg ist mittels Datenfernübertragung mit dem Organisations- und Rechenzentrum der Universität verbunden. Ein Personalcomputer wurde über Telefonkabel mit der zentralen EDV-Anlage gekoppelt. Die Physiker hatten gemeinsam mit Hard- und Software-Spezialisten des Rechenzentrums die Voraussetzungen für die Datenfernübertragung geschaffen. Das Projekt ist Bestandteil eines langfristig geplanten Ausbaus der dezentralen Zugriffsmöglichkeiten an der Universität zum Großrechner.

ADN

Entwicklung Integrierter Schaltungen bis zum Jahr 2000

Systemtechnische Anforderungen

Prof. Dr. Dietrich Eckhardt
Akademie der Wissenschaften der DDR,
Zentralinstitut für Kybernetik
und Informationsprozesse,
Institutsteil Dresden

Die moderne Mikroelektronik und Computertechnik sind zum bestimmenden Motor der Technologieentwicklung geworden. Etwa ¾ der Arbeitsproduktivitätssteigerung der hochentwickelten Industriestaaten werden heute bereits von ihnen getragen. Die Anwendung dieser Technik verändert alle Bereiche der Arbeit und der Konsumtion, insbesondere aber das Gebiet der informationellen menschlichen Arbeit, der organisatorischen und wissenschaftlich-technischen Tätigkeiten, in denen bisher keine umfassende Automatisierung möglich war. In dieser Hinsicht stehen wir am Beginn einer großen Umwälzung. Die Grundlage dafür wird mit der weiteren Entwicklung der Bauelementebasis, insbesondere der Integrierten Schaltungen gelegt. Im folgenden sollen einige Trendaussagen und Entwicklungszusammenhänge zu diesem Problemkreis thesenhaft angegeben werden. Dabei sind hier wichtige Gebiete, wie z. B. die Kommunikationstechnik, nur ganz knapp und daher nicht in der ihnen zukommenden relativen Bedeutung einbezogen.

1. Einführung

Haupttrends

Es werden folgende Haupttrends vorangestellt:

⓪ *Grundlage* aller Entwicklungstrends der Mikroelektronik und ihrer Anwendungsgebiete ist die im Integrationsgradwachstum sichtbare IC-Technologie-Entwicklung; dabei wurden folgende DRAM-Maximalwerte erreicht bzw. prognostiziert: 1962: 10, 1972: 10000, 1982: 1 Mio, 1992: 100 Mio, 2000: 1000 Mio (in Transistoren je IC). Bei Logik-IC ist die maximale Komplexität bis ⅓ der DRAM-Dichte. Dabei werden neue Technologien, Basismaterialien und Schaltungstechniken entwickelt. Diese Entwicklung führt zur Höchstintegration.
Die Hauptinnovationen liegen in der Anwendung.

① *IC-Trends* aus Anwendersicht: Ausschöpfung des erreichbaren Integrationsgrades (Komplexität, Geschwindigkeit) bei

- Standard-ICs (STDIC), insbesondere
– Speicher-IC
– Microcomputer-IC (Datentechnik, Steuerungstechnik)
– Signalprozessoren (Signalverarbeitung)
– Anwendungsspezifische IC-Typenspektren (Kommunikationstechnik, Konsumelektronik)
- Anwendungsspezifische (vom Anwender – Kunden der Halbleiterindustrie – entwickelte) ICs (sog. ASIC – oder auch kundenspezifische Schaltkreise genannt): entscheidendes Anwachsen ihrer Bedeutung zum
– Realisieren von Funktionen in Hardware anstelle in Software
– Systemintegration

② *Systemtrends* aus Mikroelektroniksicht: Integration zunehmend komplexerer Teilsysteme und Systeme in ICs

- Vordringen der modernen Informationstechnologien in alle Bereiche: Preisrückgang, hohe Stückzahlen, funktionelle Expansion und Leistungssteigerung, breites Leistungsspektrum
- Akzeptanz für eine große Zahl von standardisierten Subsystemlösungen der Computer-/Kommunikationstechnik Systemstandards (Hard- und Software): Integration von Subsystemen in VLSI-IC
- Neue Systemkonzepte: Vernetzung, Dezentralisierung, funktionelle Spezialisierung

– Verschmelzen von Rechen- und Kommunikationstechnik
– CAD/CAM/CAE und CIM sowie Büroautomatisierung
– spezialisierte Computer für spezielle Funktionsgebiete:
– Datenbasis/Kommunikations/Druck... – Server
– Simulations/Verifikations... – Acceloratoren
– Einsatz von Methoden der künstlichen Intelligenz (KI) ermöglichen es, zunehmend besser Expertenwissen breit zu nutzen
– Durchsetzen neuer Computerarchitekturen, wie Vektor-, Datenstruktur-, Datenflußarchitekturen, erfolgt bereits mit der derzeitigen Technik
– Durchsetzung neuer KI-Architekturen erfolgt schrittweise in den 90er Jahren

Anwendungsgebiete

Anwendungsgebiete (Tafel 1) von ICs sind international insbesondere folgende Gebiete, die auch langfristig von Bedeutung sind.
Dabei liegen derzeit die größeren Steigerungsraten neben der Rechen-/Datentechnik (Ausrüstung eines sehr großen Teiles von Arbeitsplätzen (CAD/CAM/CAE/CIM, Büroautomatisierung) mit personeller Rechentechnik und entsprechender Hintergrundtechnik, Nutzung der Rechentechnik in der Privatsphäre) insbesondere in neuen Gebieten wie der Automobilelektronik. In der Rechen-/Datentechnik, der Industrieelektronik und in der Autoelektronik ist der Anwendungszuwachs besonders mit einem starken Zuwachs des Einsatzes anwendungsspezifischer hoch- und höchstintegrierter ICs verbunden.

Tafel 1 Hauptanwendungsgebiete von ICs im Jahre 1984

Anwendungsgebiete	Marktanteil in %	Jahreszuwachs in %
Rechen-/Datentechnik	38	13
Kommunikationstechnik	20	7
Industrieelektronik	13	9
Konsumelektronik (klassische)	26	3
Automobilelektronik	3	15

Insgesamt gilt als langfristige Steigerungsrate wertmäßig etwa 25 % je Jahr.

IC-Typenspektrum

Standard-Schaltkreise

Besonders – in Zukunft noch zunehmende – Bedeutung besitzen **hochintegrierte Standard-IC:**

– Speicher-IC bis 40 %
– Microcomputer-IC bis 30 %
– Interface-IC bis 20 %

Hinzu kommen spezialisierte Typen von VLSI-ICs, die komplexe Standardlösungen (international standardisiert oder zumindest sogenannte Industrie-Standards für wichtige Teil-Systeme) realisieren. Diese anwendungsspezifischen IC-Typenspektren von STDIC sind oft aus Kundenwunschlösungen (ASIC) hervorgegangen.
Die Bedeutung der klein- und mittelintegrierten IC-Reihen sinkt relativ ab (unter 20 %), obwohl der Bedarf vorerst absolut noch ansteigt. Ihre besondere Bedeutung wird auch in der Zukunft folgende Aufgaben betreffen:

– Interface-Bildung/Treiber/Empfänger/Dekoder/Multiplexer
– lokale kleine Logik-Komplexe (— ASIC, insbesondere für High-Speed-Anforderungen;
— Supercomputer (ECL/CML- und GaAs-Gate-Array)
— Mainframes, Superminis (ECL, 100K, ALS, ACL).

Mit dem weiteren Anwachsen des Integrationsgrades wird bis zum Jahr 2000 keine prinzipielle Verschiebung der Anteile erfolgen, aber eine wesentliche Steigerung der Stückzahlen, bis zu etwa 30 % pro Jahr, erreicht.
Speicher-IC werden in allen wesentlichen Anwendungen etwa im gleichen anteiligen Umfange wie bisher benötigt. Es werden Zunahmen der Arbeitsspeicher erforderlich, die etwa dem Integrationsgradwachstum entsprechen. Dabei sind die ebenso steigenden Anforderungen an die Speichergeschwindigkeit zu realisieren.

Darüber hinaus sind neue Forderungen in bezug auf Datensicherheit und Datenerhalt zu lösen. Speicher-IC werden sich von der Komplexität her zu spezialisierten Systemen entwickeln. Mit dem weiteren Komplexitätswachstum wird sich die Integration von Speicher-Controllern vollziehen. Dabei werden nicht nur einfache Zugriffsfunktionen (Schutz), sondern komplexe Datenidentifikationsfunktionen (Aufstellung von Struktur- und abstrakten Datentypen) realisiert. Der Speicher-Controller wird zum Speicher-Prozessor.

Microcomputer-IC (MC-ICs werden in den nächsten 10 Jahren noch wesentlich komplexer. Der Trend zu Systemfamilien mit einem hohen Maß an Hard- und Softwarekompatibilität wird zu sehr effektiv anwendbaren Systemen geführt:

- Controller (4 bis 32 Bit),
- universelle Microcomputer-IC-Systeme (8, 16, 32 Bit),
- Signalprozessoren (20 bis über 40 Bit).

Dabei führen der wachsende Integrationsgrad und die ebenso steigende Arbeitsgeschwindigkeit zu völlig neuen Lösungsmöglichkeiten für den Systementwurf.
Es wird der Entwicklungsspielraum für qualitativ neue Technikentwicklungen geschaffen. Beispiele können sein:

- Supercomputerverarbeitungsleistung mit Microcomputersystemen in universellen Rechnersystemen:
— Uniprozessoren (größer 100 MIOPS)
— Multiprozessorsysteme (größer 1000 MFLOPS)
- KI-Systemkomponenten / Systemkomponenten.

Systeme

- spezialisierte Systeme mit sehr hohen Leistungsanforderungen (Simulation/DRC-Acceleratoren, Datenbasis-/Druck-/Zeichen-/Kommunikations-Server ...).

Die Hauptentwicklung bei den Typenspektren von anwendungsspezifischen Standard-ICs wird ebenfalls in der vollen Nutzung des Integrationsgrad- und Geschwindigkeitswachstums liegen.
Ihr Anteil wird mit der weiteren Entwicklung der Anwendungsbreite von komplexen kommerziellen Systemen, die in vielen Bereichen (z. B. bei Personalcomputern) die Dimensionen der Konsumelektronik erreicht, über diese Systemstandards entscheidend ansteigen.
Die weitere Entwicklung bei den klein- und mittelintegrierten Standard-IC-Reihen wird sehr davon abhängen, wie sich kombinierte MOS-Bipolar-Technologien entwickeln und durchsetzen lassen (BICMOS). Außerdem hängt sie von der schritthaltenden Entwicklung zu VLSI-IC-adäquaten Träger- und Verbindungstechnologien ab. Ihre besondere Bedeutung für High-Speed-Technik bleibt bestehen.

ASIC

Eine sehr rasch zunehmende Bedeutung kommt den von den Anwendern selbst entworfenen anwendungsspezifischen ICs zu, den sogenannten ASIC (Application Specific Integrated Circuit). Ihr Anteil am Wertvolumen des Weltmarktes wird bis 1995 auf einige 10% ansteigen. Ganz besonders bedeutsam ist aber der zu erwartende Anstieg am Typenumfang (dort werden sie den überwiegenden Anteil darstellen) und ihre Bedeutung für das Leistungsniveau kompletter Geräte- und Systemlösungen. Komplexe ASICs ermöglichen insbesondere die Realisierung von Systemen aus VLSI-IC mit

- zugeschnittenen, anwendungsoptimierten Lösungen (auch und besonders in Niedrigststückzahlgebieten)
- Nachbau-/Know-how-geschützten Lösungen (auch und besonders in Hochstückzahlgebieten)

Tafel 2 ASIC-Typen

Typ	Integration	Einsatzziel	Anwendungstyp
PL (programmierbare Logik)	klein	Ersatz SSI/MSI-Logik*	anwenderprogrammierbar (PAL, IFL)
GAL (Gate-Array-Logik)	klein, mittel	Logik-Integration*	IO- und Bus-intensive Architekturen
SCL (Standard-Cell-Logik)	mittel, hoch	Logik-, System-Integration*	Speicher- und spezielle Blöcke auf Chip
MCL (Macro-Cell-Logik)	hoch	System-Integration*	komplexe Baublock-Architekturen
CCD (Conventional Custom Design)	sehr hoch	System-Integration*	komplexe irreguläre Architekturen

* Anwendung von Siliconcompilern wird üblich

ASICs ermöglichen mittels VLSI die Integration spezialisierter komplexer Systeme/Teilsysteme.
In diesem Sinne und über das Erfordernis der Bereitstellung adäquater ASIC-Entwurfssysteme sowie entsprechender Rechentechnik wird deutlich, daß der ASIC-Einsatz mit der VLSI-Technik stark an Bedeutung zu gewinnen beginnt.
Die verfügbaren und in Entwicklung befindlichen ASIC-Entwurfssysteme erlauben bereits eine sehr effektive Entwurfsarbeit. An der Weiterentwicklung der Entwurfssysteme wird – wie an der Weiterentwicklung der Herstellungstechnologien – mit hoher Intensität gearbeitet. Dabei soll erreicht werden, daß der Entwurfsaufwand trotz ständig weiter steigender IC-Komplexität nicht ansteigt und daß Systementwickler ASICs selbst entwikkeln können. Entwurfssysteme für VLSI-ASICs sind „Systementwurfssysteme". Z. Z. haben sich die in Tafel 2 aufgeführten prinzipiellen ASIC-Typen durchgesetzt. Die Vielfalt wird insbesondere bei hochkomplexen ASICs steigen.

ASICs sind daher charakteristisch für

- Systementwickler bei der Schaffung optimaler Systemarchitekturen zur Ergänzung des Standard-IC-Sortimentes (Rechen-/Daten-, Kommunikations- und insbesondere Industrieelektronik),
- weitgehend komplette Systemintegration mit ASICs in Hochstückzahlanwendungen (Konsumelektronik, Auto).

Die weitere Entwicklung wird erst der eigentliche (erwartete) Lebenszyklus der ASICs sein; die ASIC-Ära ist gegenwärtig dabei, besonders breitenwirksam zu werden. Diese Entwicklung ist daran gebunden, daß

- effektive CAD-Systeme verfügbar sind und
- die Anwendungsprobleme im Sinne von Systemarchitektur und IC-Spezifikationen aufbereitet sind.

2. Computertechnik

Die Computertechnik ist in vielerlei Hinsicht zu einem bestimmenden Motor der allgemeinen Technik- und damit auch der gesellschaftlichen Entwicklung geworden. Ihre Entwicklung ist hochgradig an die Entwicklung der Schaltkreisbasis gebunden.

Generationen und Klassen der Computertechnik

Die beiden ersten Rechnergenerationen auf der Basis von Elektronenröhren (bzw. Relais) und von diskreten Transistorbauelementen und Dioden waren insbesondere für wissenschaftlich-technische/ökonomische Rechnungen (in zentralisierten Rechenzentren) einsetzbar.
Mit der Einführung der IC-Reihen (Integrationsgrad SSI/MSI) gelang es in den 60er Jahren, die ZVE (CPU) und den Großteil der übrigen Logik mittels komplexer Bauelemente sehr viel leistungsfähiger, billiger, kompakter und zuverlässiger zu integrieren. Diese dritte Rechnergeneration prägte bereits entscheidend die im wesentlichen noch heute gültigen Systemlinien.
Die derzeitige vierte Generation nutzt weitestgehend die erreichten Fortschritte der IC-Technologien:

- VLSI-Speicher-IC
- MC-IC-Systeme
- anwendungsspezifische STDIC-Typenspektren (Realisierung von Systemstandards)
- ASIC zur optimalen Architekturimplementierung (Know-how, Geschwindigkeit)

sowie die IC-Reihen (in ihren modernen Weiterentwicklungen).

Gleichzeitig wurden neue Bauelementetechnologien für die Anwendung vorbereitet:
GaAs-IC für Super-High-Speed-Anforderungen. Es wurde das gesamte Spektrum der Geräte- und Baugruppentechnologien für alle relevanten Subsysteme entscheidend weiterentwickelt: Träger-/Verdrahtungs-/Gefäß-/Kühl-Technologien, magnetomotorische Speicher, Druck- und Zeichentechnik, Display-Technik, Kommunikationstechnik.

Zudem wurden die Softwaretechnologie und die Computerarchitekturen weiterentwickelt, so daß der Übergang in eine fünfte Generation in den 90er Jahren bereits weitgehend methodisch vorbereitet wurde.
Als 5. Computergeneration werden Systeme erwartet, die einen sehr hohen Grad von Weiter- und Neuentwicklungen auf allen relevanten Gebieten (Technologien, Architekturen, Funktionsumfang, Leistungsspektrum) repräsentieren, insbesondere mit der technischen Nutzung von Methoden der künstlichen Intelligenz (KI).
Die Nutzung von KI-Methoden zum Effektivieren der Arbeit mit dem Computer (Nutzerinterface, Expertensysteme, Wissensverarbeitung) erfordert entscheidend leistungsfähigere Speicher- und Verarbeitungsressourcen sowie entsprechende Kommunikationsleistungen. Es werden sich für diese Zwecke völlig neue Architekturen durchsetzen. Daher ist mit der Einführung solcher Techniken nicht in breiter Front, aber bereits jetzt beginnend zu rechnen. In den späten 90er Jahren wird KI-Technik allgemein anwendbar sein.
Die Computerklassen stellen eine in Leistungsvermögen, Preis, Nutzungsgesamtaufwand einerseits und Einsatzbreite andererseits gestufte Pyramide dar. Dabei ist es prinzipiell erforderlich, die gesamte Pyramide zu betrachten. Im folgenden soll nur die universell nutzbare, nicht die spezialisierte Rechentechnik wie z. B. Mikrorechnersteuerungstechnik, betrachtet werden, obwohl diese natürlich ebenfalls von großer Bedeutung ist.

Supercomputer

Supercomputer stellen in allen Belangen Spitzentechnologien dar! Sie benötigen Bauelemente und Baugruppen mit höchsten Geschwindigkeits- und Komplexitätsparametern. Sie ermöglichen ökonomisch effektiv die Ausführung von
- extrem rechenintensiven Programmen (Verarbeitungsgeschwindigkeit), bzw.
- sehr vielen parallel vorliegenden Aufträgen (Verarbeitungsdurchsatz).

Der erste Aufgabentyp (Simulation. Pattern-Verarbeitung ...) kann u. U. bei stabilen Auftragssituationen ökonomischer zu Spezialarchitekturen (VLSI-Implementierung) führen.
Der zweite Aufgabentyp erfordert die parallele Auftragsbereitstellung (große Externspeicher, flächendeckendes Kommunikationsnetz).
Supercomputer benötigen derzeit Hochleistungs-ECL- oder GaAs-GAL-IC mit Spezialgehäusen und -kühlverfahren.
Die derzeitigen Leistungsnormative liegen bei einigen 100 MIOPS und einigen 1000 MFLOPS (Vektor-Computer).

Großrechner/Mainframecomputer

Mainframes werden insbesondere in allen zur Geldwirtschaft im weiteren Sinne gehörigen Anwendungen, für Statistik und Abrechnung eingesetzt.
Sie stellen in bezug auf Logik-IC Spitzentechnologien dar: ECL-GAL.
Ihr Leistungsnormativ liegt derzeit bei 10 bis 50 MIOPS und steigt jährlich um 15%. Die Ablösung einer Generation erfolgt innerhalb von etwa 5 bis 6 Jahren.

Kleinrechner/Superminis

Besondere Bedeutung für wissenschaftlich-technische Aufgaben haben Superminis erhalten. Das sind die typischen Abteilungs-Computer der FE-Bereiche. Darüber hinaus finden sie in den klassischen Kleinrechneranwendungen, in flexiblen Steuerungen und in Warten Einsatz.
Superminis benötigen im Prozessor schnelle Standard-IC, ECL, ACL.
Die derzeitigen Leistungsnormative liegen bei 3 bis 15 MIOPS.

Personalcomputer

Die größten Wachstumsraten liegen bei den Personalcomputern (PC). Sie nutzen derzeit am effektivsten die VLSI-Fortschritte:
- VLSI-Microcomputer-Systeme
- VLSI-Speicher
- Winchester-Disk-Speicher.

Sie ermöglichen in viel höherem Maß als es mit Terminalnetzen an Groß- und Superrechnern möglich ist, die effektiven und schöpferischen Arbeiten (FE) in Form von Computerarbeitsstationen (CAD). Es sind sehr vielfältige Formen zur Vernetzung (LAN, WAN) notwendig und bereits mit der derzeitigen Technik kostengünstig realisierbar. Die derzeitigen Leistungsnormative liegen für die verschiedenen Unterklassen bei:
- Workstation (32 Bit) 1–5 MIOPS
- PC (16, 32 Bit) 0,3–3 MIOPS
- Kleincomputer, PC (8 Bit) 0,1 MIOPS

Entwicklungstrends und Anforderungen an IC-Typenspektren und Entwurf

Es sind in bezug auf die IC-Spektren folgende Entwicklungstrends festzustellen:
① Die SSI/MSI-Reihen werden für die oberen Klassen der Computersystempyramide eine bestimmte Bedeutung behalten.
② Zur Realisierung von Supercomputern werden nach 1990 neue Lösungen notwendig:
- schnellere Techniken: GaAs-IC und andere Prinzipien
- leistungsfähigere Architekturen:
— Multiprozessor/Multicomputer-Architekturen
- Vektor-, Datenstruktur-, Datenfluß-Architekturen
- KI-Architekturen (z. Z. Grundlagenforschung)
③ Zur Realisierung von Mainframes werden ECL-GAL-Techniken bis in den Zeitraum 1995/2000 ausreichend sein.
④ Zur Realisierung von Superminis sind bipolare (oder kombinierte) Technologien erforderlich. Nach 1995 werden KI-Architekturen umfassenden Einsatz finden. Es ist z. Z. offen, ob das auf der Basis von VLSI/ULSI-ASIC oder von STDIC erfolgen wird.
⑤ Die Realisierung von Personalcomputern basiert derzeit am stärksten auf dem Integrationsgrad LSI/VLSI. Mit der Durchsetzung von vollständigen VLSI-IC Spektren werden sich 32-Bit-Architekturen umfassend durchsetzen und zu einem entscheidenden Faktor der weiteren allgemeinen Technologie-Entwicklung werden.
Hauptrealisierungsmittel der Personalcomputer werden sein
- MC-IC-Systeme
- VLSI/ULSI-Speicher-IC
- ASIC zur Systemoptimierung.
⑥ Spezial-Computer werden überall dort Verbreitung finden, wo entweder sehr einfach zu programmierende Aufgaben vorliegen oder wo Standardarchitekturen als Grundlage der Programmierung genutzt werden können.
Hauptrealisierungsmittel werden VLSI/ULSI-IC sein. Dabei wird überall dort auf neue VLSI/ULSI-optimale Architekturen orientiert, wo damit kein unvertretbarer hoher Softwareaufwand entsteht.
In diesem Sinne sind die Bit-Slice- und die anwendungsspezifischen Signalprozessoren Entwicklungsbeispiele.

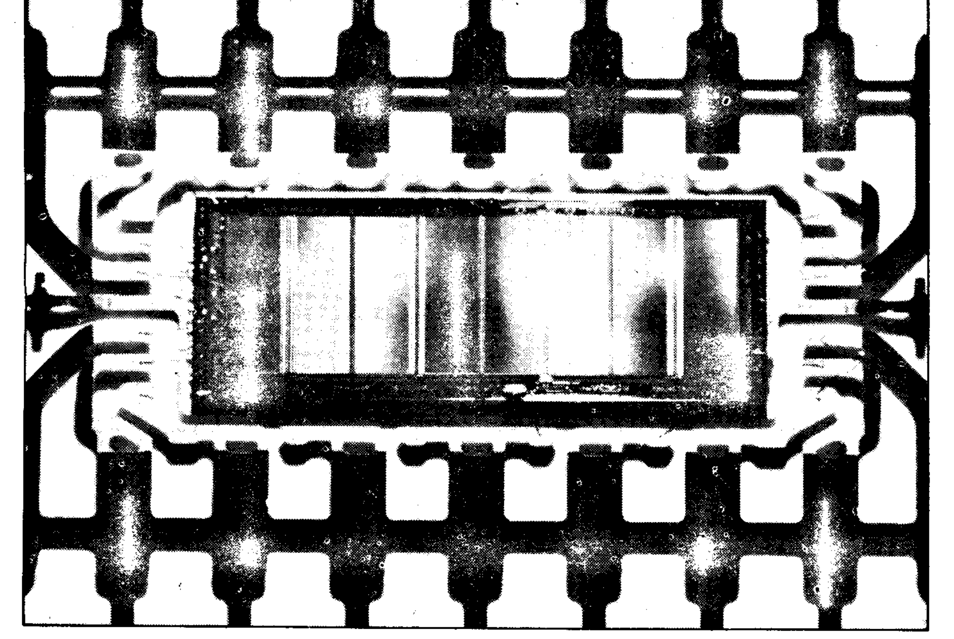
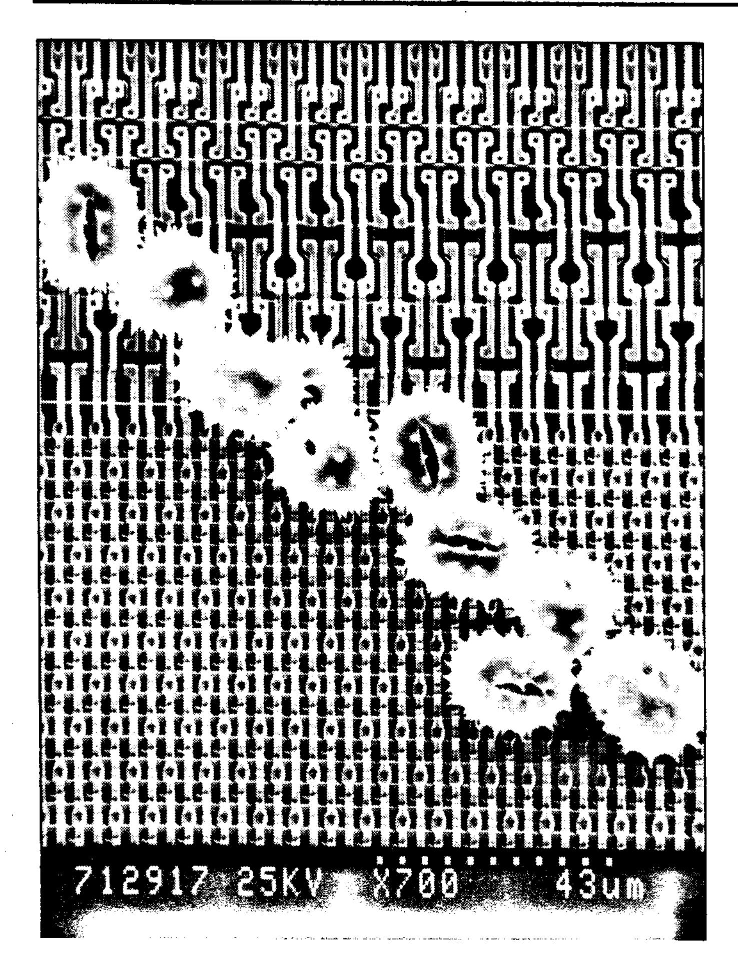
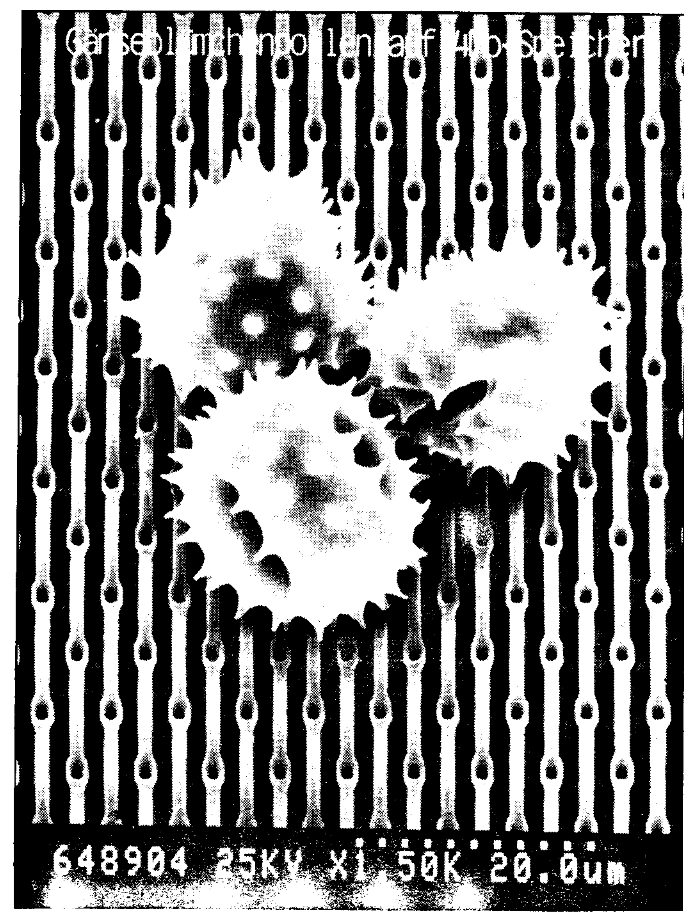


Bild 1 Dynamischer 1-MBit-Speicher mit 54 mm² Chipfläche





Bilder 2 und 3 Miniaturisierung: Gänseblümchenpollen auf 1-MBit-Speicherchip und 4-MBit-Speicherchip Werkfotos (3)

⑦ Für alle Belange der Steuerungstechnik und Industrieelektronik sind Superminis in Warten und zentralen Aufgaben erforderlich. Direkt im Steuerungsprozeß sind universelle MC-Lösungen des gesamten MC-Typenspektrums (Controller, universelle MC-IC-Systeme (8 bis 32 Bit)) notwendig und sinnvoll.
⑧ In allen Anwendungen von Analogdaten- und Signalverarbeitung sind digitale Signalprozessoren charakteristisch. Dafür sind neben den universellen DSP insbesondere auch zunehmend anwendungsspezifische DSP ökonomisch sinnvoll. Grundlage werden DSP-ASIC-Systeme sein.
⑨ Zu den derzeit noch am schlechtesten abschätzbaren Problembereichen ist die schaltungstechnische Realisierung von KI-Systemkomponenten und Systemen zu rechnen.
⑩ Haupttrends der Entwurfstechnik sind:
- Nutzung integrierter CAD/CAM/CAT-Systeme (Automation)
- ASIC-Systementwurfssysteme
- KI-Techniken

3. Digitale Signalverarbeitung

Das Gebiet der digitalen Signalverarbeitung gewinnt international in den letzten Jahren eine wesentliche Bedeutung für viele Anwendungsbereiche. Dabei werden für Signalverarbeitungsfunktionen spezialisiert optimierte Signalprozessoren breitenwirksam eingeführt. Die vergleichsweise langsamen sowie aufwendigen und daher teuren Universal-Mikrorechnerlösungen wurden damit abgelöst und viele wichtige Anwendungen (mit höheren Geschwindigkeits- und Kostenanforderungen) wurden überhaupt erst lösbar.

Digitale Signalprozessoren (DSP)

Digitale Signalprozessoren (DSP) (Digital Signal Processing/Processors) in Form von VLSI-IC unter Nutzung der bekannten Rechnerarchitekturen sind das Lösungsprinzip der modernen Signalverarbeitungstechniken. Sie erlauben den Aufbau funktionell sehr komplexer Signalverarbeitungssysteme für (vor allem Real-Time-)Anwendungen in
- Kommunikationstechnik
- industriellen Steuerungen
- Hochleistungsrechentechnik
- Graphik und Bildverarbeitung
- Sprachverarbeitung
- digitales Fernsehen.

System-Architektur

Von der Systemarchitektur her sind DSP (als Digitalteil von Signalverarbeitungssystemen) komplette digitale Computer mit auf diese Anwendungsspezifik zugeschnittener Struktur. Für die Analogsignalverarbeitung sind am DSP-Input ein A/D- und am DSP-Output ein D/A-Wandlungsbaustein erforderlich, die über serielle, oder für High-Speed-Anforderungen parallele, Schnittstellen mit dem eigentlichen DSP verbunden sind. Seltener bzw. nur bei spezialisierten DSP trifft man On-Chip-Analog-Schnittstellen an, d. h. mitintegrierte A/D- und D/A-Wandler.

Der eigentliche DSP

Der eigentliche DSP ist nur ein besonders schneller und dafür funktionell recht einfacher digitaler Datenprozessor.
Die Datenbreite und der Befehlssatz sind den Anforderungen der Signalverarbeitung in Hinsicht auf Grundfunktionen und Genauigkeit/Dynamikumfang angepaßt. Die gewählte Systemarchitektur ist von den Geschwindigkeitsanforderungen bestimmt.
DSP werden aus großen Blöcken aufgebaut, die sich sehr gut für den DSP-System- und Schaltkreisentwurf mittels ASIC-Building-Block-Systemen (Makrozellen) eignen:
- Verarbeitungseinheit mit 20 bis 48 Bit Datenbreite und mit hohem Pipeling bzw. Parallelarbeit aus
- RALU (Universal-Register, Arithmetik-Logik-Einheit)
- Verschiebe-, Multiplizier-Einheit
- Datenspeicher (On-Chip oder über Off-Chip-Interface)
- eventuell separate Konstantenspeicher.
- Programmspeicher (On-Chip-, Off-Chip-Interface) mit:
— ROM (On-Chip-ROM für spezialisierte DSP)
— PROM (Bipolartechniken, Off-Chip-PROM)
— EPROM (MOS-Techniken, Off-Chip-EPROM)
— RAM (aktuelle bzw. ladbare Daten)
- Adreßwerk für Datenspeicher
- Programmsteuerung (Mikroprogramm-Steuerung) und Adreßwerk für Programmspeicher
- Signalwert-Input/Output Interface-Einheit
- Kopplungsinterface für DSP-Multiprozessor, Array bzw. Horst-Kommunikation.

Realisierung von DSP

Die Realisierung der DSP erfolgt mit drei grundsätzlichen Realisierungswegen, als
- universelle Single-Chip-DSP (VLSI-IC, kompakteste Lösung)
— ist seit etwa 5 Jahren technisch möglich/üblich
— im allgemeinen ohne On-Chip-Analogteil
- PCB mittels DSP-Building-Block-IC-Systemen (LSI/VLSI-IC Eignung für spezielle Anforderungen/Anpassung),
— im allgemeinen als Ergänzung von VLSI-Bit-Slice-Systemen (RALU und universellen Mikroprogramm-Steuereinheiten, Support-IC)
— mit A/D- und D/A-Wandler-Funktionsblöcken (IC)
— mit VLSI-Multiplizier-Funktionsblöcken (IC), wobei die Systemgeschwindigkeit entscheidend von der Verarbeitungsparallelität bestimmt wird
- anwendungsspezialisierte DSP (VLSI-IC)
— feste, spezialisierte DSP-SC-IC, zunehmend mit On-Chip-Analogteil
— ASIC-Building-Block-Systeme von komplexen Zellen (Makrozellen) für flexible Auslegung innerhalb bestimmter Anwendungsgebiete.

DSP-Entwicklungstechnik

Zur Realisierungsunterstützung werden DSP-Entwicklungssysteme und DSP-Software angeboten. Das sind vor allem folgende Systeme
- Entwicklungstechnik für universelle SC-DSP
- Entwicklungstechnik für VLSI-Bit-Slice-Systeme und deren Erweiterungen für DSP
- Entwicklungstechnik für Multiplizier-IC-Systeme
- ASIC-Entwurfssysteme als Entwicklungstechnik für anwendungsgebietsspezifische SC-DSP (auf der Basis von ASIC-Building-Block-Systemen).

Realisierungstechnologien für DSP

Als Realisierungstechnologien sind infolge der hohen Forderungen
- Systemkomplexität: 30000 bis 600000 Transistorfunktionen/Chip und
- Systemgeschwindigkeit: 3 bis 30 MFLOPS vor allem eingeführt
- MOS-Technologien für VLSI-IC, insbesondere als SC-DSP
— NMOS bis vor 3 Jahren (Einführungsdatum)
- EPROM-Speicher (On-Chip) für änderbare DSP
— CMOS (2 bis 1 μm) für DSP der 2. und 3. Generation
- Bipolartechnologien für High-Speed-Building-Block-Systeme.

Infolge der großen Einsatzwachstumsraten und der den DSP-Anforderungen zunehmend entgegenkommenden IC-Technologieentwicklungen ergeben sich folgende Trends
- CMOS wird für einen Großteil der einfacheren Anwendungen und mit maximalem Integrationsgrad eingesetzt:
— universelle, spezialisierte und ASIC-Building-Block-SC-DSP
— allgemeine Building-Block-DSP-Systeme für schnelle spezialisierte DSP in Verbindung mit universellen Bipolar-IC

Dietrich Eckhardt, Jahrgang 1937, ist Dresdner. Er studierte an der TU Dresden Hochfrequenztechnik und Luftfahrtgeräte und diplomierte 1961 mit einem Radartechnikproblem. Von 1967 an war er Oberassistent (Schaltungstechnik) an der TU Dresden, ab 1970 Dozent (Elektronische Systeme) an der Ingenieurhochschule Dresden. Von 1975 bis 1979 arbeitete er im Kombinat Robotron. 1979 wurder er ordentlicher Professor (Schaltungsentwurf) an der TH Karl-Marx-Stadt, 1981 stellv. Direktor des Zentralinstituts für Kybernetik und Informationsprozesse der Akademie der Wissenschaften der DDR. Bereichsleiter (Schaltungssysteme und Systementwurf). Er bearbeitet Aufgaben zum Systementwurf höchstintegrierter Schaltkreise.

— es sind hoch parallele/aufwendige Architekturen zunehmend besser/billiger realisierbar: insbesondere On-Chip- und Off-Chip-Array-Strukturen
- BICMOS wird schnell, zunehmend in allen Anwendungen Einsatz finden, die höhere Forderungen an Input/Output-Flexibilität und an Arbeitsgeschwindigkeit realisieren müssen
- Für High-Speed-Spezialanwendungen, die nicht übergangen werden können, sind in den 90er Jahren GaAs-DSP zu erwarten.

Grundsätzliche Architektur

Die grundsätzliche Architektur von DSP ist üblicherweise nicht die für Universalcomputer charakteristische Princetown / v. Neumann-Architektur sondern die Harvard-Architektur, die den Daten- und den Programmbereich im Speicher trennt. Darüber hinaus werden sich mit den Möglichkeiten des Integrationsgradwachstums neue, aufwendigere und damit leistungsfähigere Architekturen einführen:
- Parallelprozessoren
- systolische Prozessoren
- Datenflußarchitekturen.

Anwendungen von DSP

Über die oben angegebenen prinzipiellen Anwendungsgebiete
- Kommunikationstechnik
- Industrielle Steuerungen
- Hochleistungsrechentechnik
- Graphik und Bildverarbeitung
- Sprachverarbeitung
- Digitales Fernsehen
hinaus sind einige Erläuterungen angebracht.

Die digitale Signalverarbeitung ist seit etwa 30 Jahren in der Militärelektronik in vielen Systemen unabdingbar.
Es erfolgte mit den Mitteln der modernen Mikroelektronik eine sehr bedeutsame Entwicklung, die anfangs aus Kostengründen nur sehr eingeschränkt, inzwischen aber zunehmend der kommerziellen und auch der Konsumelektronik zugute kommt.
Entscheidend für optimale Lösungen von DSP-Anwendungen ist die funktionelle Spezifikation mit hardwareeffizienten Algorithmen und Programmen, d. h. bei Ausnutzung architekturaler Parallelismen, im Rahmen der verfügbaren Architekturen.
Über die elementaren Signalverarbeitungsfunktionen hinaus:
- Filterung
- Fourier-Transformation
sind die Anwendungen zu sehen in der:
- Kommunikationstechnik in
— ISDN-Funktionen: Subscriber Terminal Equipment, Two-Wire-Full-Duplex, Low-Bit-Rate-Coder, Transcoder
— Designer Chips
— Fernsteuerung und Diagnose; Remote Maintenance, Monitoring and Diagnostics
- Industrielle Steuerungstechnik in
— Analog-Daten Erfassung und Verarbeitung
- Robotik: Steuerungs-Systeme mit speziellen Image-Processing-IC, Gray-Scale und anderen spezifischen Image-Signal-Processing-IC
— CIM, Gütesicherung: Vision-Inspection-Systems
- Hochleistungsrechentechnik in
— Laser-/Radar-/Sonar-Technik
— Computertomographie
— Arrayprozessoren
- Graphik und Bildverarbeitung in
— Graphik-/CAD-Systemen
— Bildverarbeitung mit Scenen-Analyse, Datenkompression, Bilderganzung und Restoration und Real-Time-Systemen.
- Sprachverarbeitung in
— Sprach/Signalerkennung, adaptive Entzerrung, Echounterdrückung, lineare Prädiktion, lineare Störungsunterdrückung
— Sprach-Synthese/Ausgabe/Eingabe
- Digitales Fernsehen (DTV) in
— VHF-, HF-, VF-Signalverarbeitung
— Steuerung
— Speicherung
— Display.

4. Entwurfssysteme

Erreichter Entwicklungsstand je Entwurfssystem

Entwurfssysteme für die Entwicklung von ICs haben international einen sehr hohen Leistungsstand erreicht. Sie repräsentieren die Leistungsspitze von CAD/CAM/CAT-Systemen.
Es haben sich dabei durchgesetzt:
- IC-Entwurfssysteme der Halbleiterindustrie
- System- und IC-Entwurfssysteme der Geräteindustrie
(Orientierung auf Architektur-Implementierung, ASIC)
- ASIC-Entwurfssysteme der Halbleiter-, der Geräte- und der CAE-Systeme-Industrie (Lieferanten für CAD/CAM/CAT-Technik/Komponenten)

Anforderungen an die CAD-Rechentechnik

Die volle Beherrschung des Integrationsgrades VLSI erfordert eine qualitativ neue Generation von Entwufssystemen. Das betrifft die Belange aller Grundkomponenten eines Entwurfssystemes:
- Basis-Rechentechnik
— Rechnerpyramide, speziell CAD/CAM/CAT-Technik
— Basissoftware
- Basis-CAE-System, integrierte Systemumgebung
— Datenbasis
— Methodenbasis, einschließlich KI- und Lernkomponenten
— Nutzer-Interface
- Tool-System (Entwurfsprogrammsysteme Softwarewerkzeuge)
— CAD (Produkt-Entwurf/Synthese)
— CAM (Fertigung)
— CAT (Testung: Produkt, Herstellungsprozeß).

Tafel 3 Beispiel der Entwurfshierarchie des SRC-Forschungsprojektes der Carnegie-Mellon-University (USA)

Niveau	Name	Beschreibungs-Sprache	Ziele/Operatoren/Beispiele
0	Aufgabe	(Manual)	Produkte, Markt-Position, Leistungsparameter, Applikationen
1	Architektur	(PMS)	Systemblöcke; Prozessoren, Controller, Busse
2	Funktion	ISPS	Funktionen, Implementierung unabhängig; Operationen, in Form von Programmzeilen
3	Logik-Block	DIF	Datenpfad-, Steuerblockstruktur, Steuerablauf; ALU, REG, MUX
4	Logik-Bit	DIF	Gatter-Logik-Struktur/Schaltung; Logikelemente, Signalverbindungen
5	Elektronik	DIF	Transistorstruktur/Schaltung; Transistoren, Verbindungen
6	Layout		Maskengeometrie; Ebenen, Figuren/Polygone/Punkte
7	Prozeß		

Eine besondere Bedeutung kommt der CAD-Rechentechnik zu.
Einfachste ASIC-Systeme in der Geräteindustrie sind bereits mit entsprechend ausgestatteten 16-Bit-PC realisierbar. Zumindest das Erlernen des Umgangs mit leistungsfähigen Systemen kann bereits an PC-Systemen erfolgen.
Die üblichen VLSI-ASIC-Systeme sind ohne 32-Bit-Arbeitsstationen nicht zu realisieren. PC dabei zusätzlich einzusetzen, kann sinnvoll sein (Software-, Benutzungskompatibilität). Im System-Hintergrund sind leistungsfähigere Ressourcen erforderlich (Supermini, Mainframe-, Supercomputer).
Darüber hinaus sind zunehmend spezialisierte CAD-Ressourcen wie Accelerator- und Server-Systeme im Einsatz, die im Rahmen von leistungsfähigen LAN-Architekturen (z. Z. Ethernet) eingebunden werden. Damit wird in hohem Maß Inselfähigkeit und Flexibilität erzielt. In der damit vorgezeichneten Weise wird sich die weitere Effektivierung der CAD-Rechentechnik vollziehen.
Sowohl in der Geräte- als auch in der Halbleiterindustrie sind „inhomogene" CAD-Rechnersysteme üblich, die die gesamte Pyramide umfassen. Letztendlich sind das auch Ergebnisse einer langen betrieblichen Entwicklung und der Anhäufung von CAD/CAM/CAT-Arbeitstechnologien (Programmsystemen, Daten, Erfahrungen), die nicht ohne weiteres übertragbar sind.

Aufgaben der Forschung

VLSI-Entwurfssysteme integrieren zunehmend vollständig die Mittel/Tools zur durchgängigen Rechnerunterstützung während des gesamten Entwicklungs- und Produktbetreuungsablaufes von VLSI-IC. Sie sind daher integrierte CAE- bzw. CAD/CAM/CAT-Systeme. Ihre Aufgabe ist zunehmend nicht nur die Sicherung der Entwurfsoptimalität und Fehlervermeidung, sondern (insbesondere bei ASIC) die Produktivität des Entwurfs-/Entwicklungsprozesses.
Gleichzeitig kommt es auch darauf an, die Durchgängigkeit des Entwurfs-/Entwicklungsprozesses über alle Beschreibungs-/Abstraktions-Levels rechnergestützt zu vollziehen. Besondere Schwierigkeiten bietet dabei derzeit noch der Vorbereitungsprozeß für IC-Entwürfe/Entwicklungen. Dafür gibt es international erste Testsysteme in Forschungseinrichtungen (Tafel 3). Es wird notwendig sein, für die oberen Niveaus des Entwurfseinstieges und der IC- bzw. System-Spezifikation CAD-Tools zu schaffen, die der IC-Komplexität und der der erreichbaren eigentlichen Entwurfseffektivität entsprechen. Das ist eine ständig mitwachsende Aufgabe.
Eine längerfristig ganz entscheidende Weiterentwicklung der CAE-Systeme für den VLSI/ULSI-IC-Entwurf ist die adäquate Einbeziehung von KI-Methoden. Das betrifft die Effektivierung der Entwurfsprozesse in mehrerer Hinsicht.

An KI-Komponenten kommen insbesondere in Frage

- intelligentes User-Interface (NLI (Natural-Language-Interface), Graphik-Interface)
- intelligente CAE-Systemumgebung
- Expertensysteme innerhalb des Projekt- und des Datenbasismanagements
- Expertensysteme innerhalb vieler CAD-Tools.

Berichtigung

Im Beitrag *Ladeadressenanzeige für KC 85/2* in MP 4/87, S. 126 muß die 2. Zeile des Programmausdrucks richtig lauten:
C5 CD ...
Wir bitten, den Fehler zu entschuldigen. *MP*

16-Bit-Single-Board-Computer SBC 8086

Prof. Dr. Bernd-Georg Münzer,
Tomasz Stachowiak
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik

Vorbemerkung

16-Bit-Mikroprozessorsysteme gewinnen in Lehre und Forschung an den Hochschulen der DDR eine wachsende Bedeutung, so daß Lösungen erforderlich werden, die den Studierenden den unmittelbaren Zugang zur applikativen Praxis ermöglichen.
Die für die Aus- und Weiterbildung relevanten 16-Bit-Mikroprozessorsysteme basieren auf den Prozessoren K 1810 WM 86 (I8086) und U 8000 (Z 8000).
Im vorliegenden Beitrag wird der Single-Board-Computer SBC 8086 vorgestellt /1/, /2/ der auf der Basis des Mikroprozessorsystems K 1810 WM 86 (8086) aufgebaut wurde /3/.
Der Einplatinen-Rechner soll vorrangig für die Lehre eingesetzt werden und in Form einer Zusatzleiterkarte mit den Abmessungen 230 mm × 250 mm an jeden 8-Bit-Bürocomputer über eine parallele oder serielle Schnittstelle anschließbar sein (Bild 1). Der über einen Steckverbinder verfügbare Systembus ermöglicht es aber auch, den SBC 8086 mit zusätzlichen Modulen (z. B. Speicher) zu erweitern, um einen Einsatz als OEM-Rechner für die Prozeßautomatisierung zu realisieren.

Struktur des SBC 8086

Der SBC 8086 besitzt einen modular erweiterbaren Aufbau mit gepuffertem Systembus (Bild 2) und setzt sich aus folgenden Baugruppen zusammen:
- Zentrale Verarbeitungseinheit mit den Schaltkreisen
CPU 8086
Clockgenerator 8284
Buscontroller 8288
- Arbeitsspeicher mit 4 KByte EPROM und 128 KByte DRAM
- Peripheriebaugruppe mit
2 seriellen Schaltkreisen 8251
2 parallelen Schaltkreisen 8255
1 Timer 8253 und
1 Interrupt-Controller 8259A.

Zentrale Verarbeitungseinheit

Der Mikroprozessor K 1810 WM 86 wird im SBC im Maximum-Modus betrieben. Der Systemtakt von 5 MHz, Tastverhältnis 1 : 3, wird vom Clockgenerator 8284 bereitgestellt, und dieser synchronisiert auch das RESET- und READY-Signal mit dem Systemtakt.
Zu Beginn jedes Buszyklus werden von der CPU die Statussignale S0, S1, S2 ausgege-

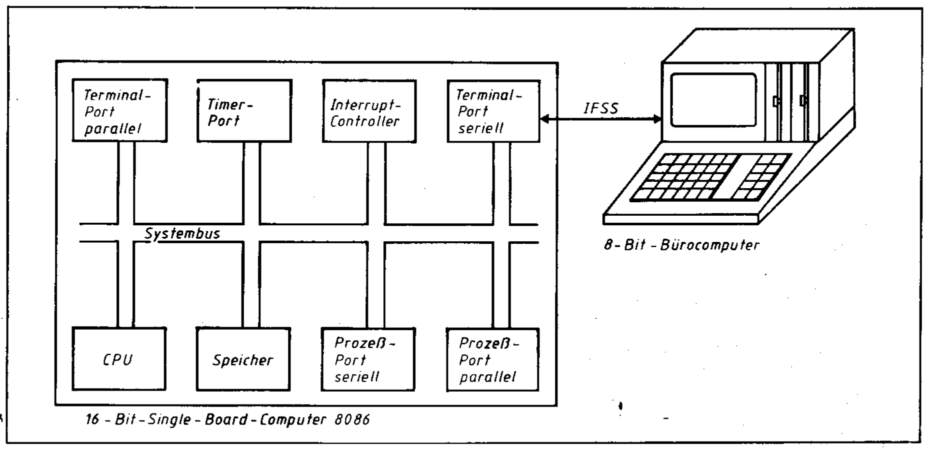


Bild 1 16-Bit-Mikrorechnerkonfiguration

ben, die vom Buscontroller 8288 dekodiert werden und der dann die Signale des Steuerbus generiert (Bild 3). Gleichzeitig mit den Statussignalen wird die aktuelle Adresse ausgegeben. Der Transfer der Daten und Adressen erfolgt zeitmultiplex auf dem Bus AD 0 ... AD 15. Die Adressen A 16 bis A 19 werden wiederum mit anderen Statussignalen multiplex geführt. Die Selektion in den Systemdaten- bzw. Systemadreßbus erfolgt mit Hilfe der Steuersignale ALE und DEN, wobei die Adressen mit ALE in den Oktal-Latches gespeichert werden.

Arbeitsspeicher

Der Adreßraum des 8086 von 1 MByte wird physisch in zwei Banks mit je 512 KByte aufgeteilt (EVEN-Bank, ODD-Bank). Die Even-Bank umfaßt alle geraden Adressen und ist mit dem niederwertigen Teil des Systemdatenbus (DB0 ... DB7) verbunden. Dementsprechend umfaßt die ODD-Bank alle ungeraden Adressen und ist mit dem höherwertigen Teil des Systemdatenbus (DB8 ... DB 15) verbunden. Die Selektion erfolgt mit Hilfe der Signale A0 und BHE.
Der 4-KByte-Festwertspeicher (2xEPROM U2716) befindet sich im obersten Teil des physischen Adreßraumes bei FF000H ... FFFFFH und beinhaltet den Bootstrap-Loader, der nach RESET aufgerufen wird.
Der Operativspeicher ist ein DRAM-Modul vom Typ ZERO-WAIT-STATES und umfaßt die ersten 128 KByte des physischen Adreß-

Prof. Dr. sc. techn. Bernd-Georg Münzer (46) diplomierte 1964 als Physiker an der Technischen Universität Dresden; von 1964 bis 1978 war er auf dem Gebiet der Entwicklung von integrierten Schaltkreisen im Institut für Mikroelektronik Dresden tätig. 1969 Promotion A; 1978 bis 1983 Delegierung an die Technische Universität Dresden, Sektion Informationstechnik, mit Promotion B zu Interfacetechniken an Mikrorechnern; 1984 Berufung zum Ordentlichen Professor.
Prof. Münzer ist Leiter des Wissenschaftsbereiches Mikrorechentechnik/Schaltungstechnik, Sektion Technische Elektronik, an der Wilhelm-Pieck-Universität Rostock.

Tomasz Stachowiak (28) studierte an der Technischen Universität Dresden und diplomierte 1982 zum Thema „Mehrrechnerkopplung mittels transparentem DMA-Verfahren“. Nach einer einjährigen Tätigkeit als wissenschaftlicher Mitarbeiter im WTZ Holz in Dresden arbeitete er bis 1986 in einem Industrie-Institut in Lodz (VR Polen). 1986 begann er eine Aspirantur an der Wilhelm-Pieck-Universität Rostock auf dem Gebiet der Multirechnersysteme.

Bild 2 Architektur des Single-Board-Computers SBC 8086

raumes (00000 ... 1 FFFFH mit 16 DRAM-Schaltkreisen 64 KBit × 1 U 2164).
Die 8086-CPU unterstützt nicht den Refresh für die dynamischen Speicher, so daß der DRAM-Modul eine Logik enthält, die die Steuersignale RAS und CAS beim Adreßmultiplex für Lese- und Schreiboperationen generiert und auch einen Auto-Refresh gewährleistet.

Peripherie-Modul

Der Peripherie-Modul gliedert sich wie beim Speichermodul in eine EVEN- bzw. ODD-Bank. Die EVEN-Bank stellt das Terminal-Interface dar und ist adreßkompatibel zum Debug-Monitor 957B /4/, /5/. Die Datenleitungen der Interface-Schaltkreise sind zu einem internen I/O-Datenbus verbunden, der über Bustreiber direkt an die 8086-CPU geführt wird.

- *EVEN-Peripherie*

Terminal-Port-Seriell:
– Daten-duplex mit Modemsteuersignalen und Sende-/Empfangstakt
– EVEN-USART 8251, Portadressen D8H, DAH

Terminal-Port-Parallel:
– 2 parallele Ein- oder Ausgabe-Ports je 8 Bit (PA und PB) mit Handshaking-Signalen
– EVEN-PPI 8255, Portadressen C8H, CAH, CCH, CEH

Timer-Port:
– 3 Zähler-/Zeitgeber-Kanäle, Kanal 2 als Taktversorgung für EVEN-USART
– EVEN-TIMER 8253, Portadressen D0H, D2H, D4H, D6H

Interrupt-Controller:
– vektorisierte Interruptbehandlung
– 8 priorisierte Interruptebenen
– 11 über Wickelmatrix geführte Interruptquellen von EVEN/ODD-USART, EVEN/ODD-PPI, EVEN-TIMER und extern über Systembus
– EVEN-PIC 8259A, Portadressen C0H, C2H

- *ODD-Peripherie*

Die ODD-BANK repräsentiert das Prozeß-Interface und ist über Bustreiber an den höherwertigen Teil DB8 ...DB15 des Systemdatenbusses geführt. Die ODD-Interface-Schaltkreise haben jeweils die nachfolgend ungeraden Adressen der entsprechenden EVEN-Schaltkreise, so daß neben dem Byte-Transfer auch der Wort-Transfer möglich ist.

Prozeß-Port-Seriell:
– Datentransfer wie beim Terminal-Port
– ODD-USART 8251, Portadressen D9H, DBH
– Taktversorgung vom EVEN-Timer, wickelbar

Prozeß-Port-Parallel:
– Datentransfer wie beim Terminal-Port
– ODD-PPI 8255, Portadressen C9H, CBH, CDH, CFH
– Die Ports PA oder PB von EVEN- und ODD-PPI können paarweise als 16-Bit-parallele Schnittstellen für Worttransfer benutzt werden.

Die Daten- und Steuersignale der Terminal- und Prozeßports sind direkt an Steckverbinder geführt. Durch zusätzliche Interface-Adapter können Standard-Schnittstellen realisiert werden (IFSS, V.24, IFSP, Centronics).

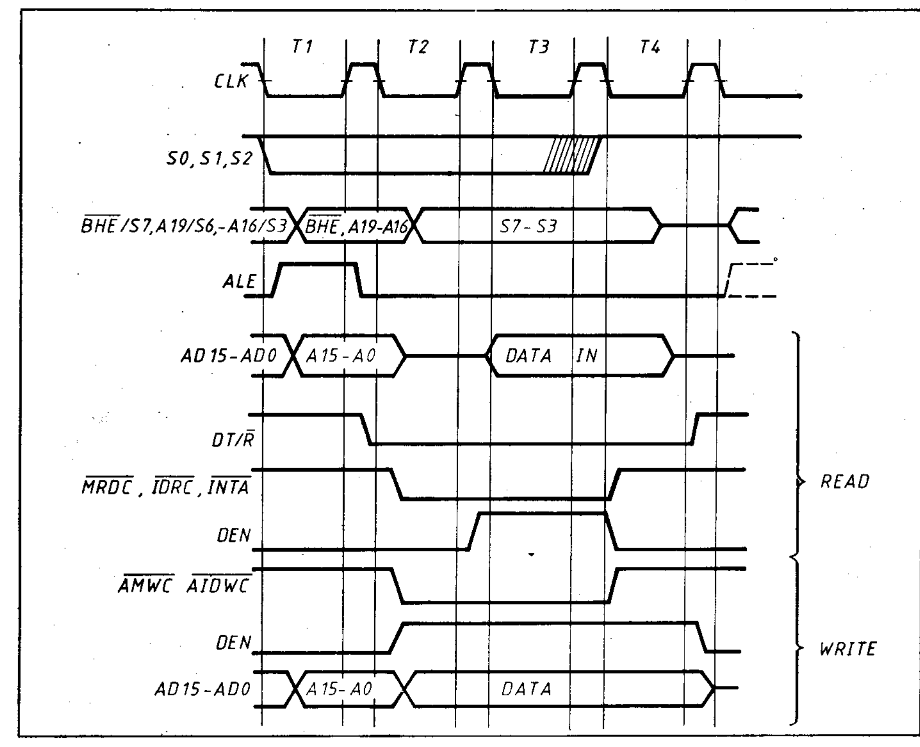
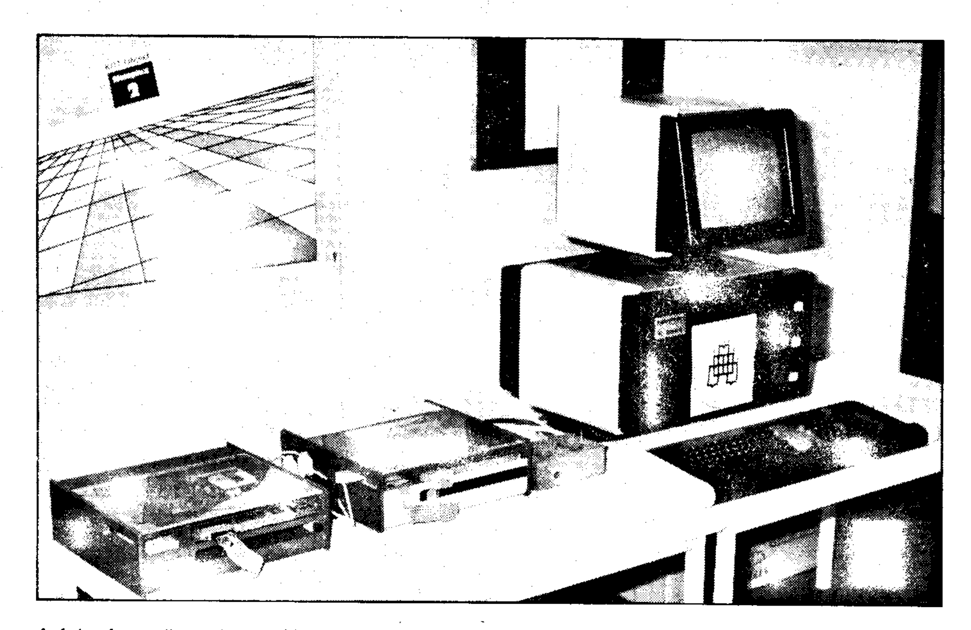


Bild 3 Buszyklus des 8086



Auf der Ausstellung des 12. Mikroelektronik-Bauelemente-Symposiums 1987 in Frankfurt (Oder) gezeigter Mikrorechnerarbeitsplatz der Wilhelm-Pieck-Universität Rostock, bestehend aus einem 8-Bit-Mikrorechner und den 16-Bit-Single-Board-Computern SBC 8086 und SBC 8000

Der 8-Bit-BC als intelligenter Peripherie-Controller

Der SBC 8086 besitzt keine eigene Konsole, so daß eine Kombination des 16-Bit-Mikrorechners SBC 8086 mit einem 8-Bit-Bürocomputer realisiert wurde. Die Kopplung erfolgt über die seriellen/parallelen Terminalports des SBC 8086 mit den entsprechenden Schnittstellen des 8-Bit-Rechners.
Durch diese Lösungsvariante wird der 8-Bit-Rechner mit seinen peripheren Ressourcen Tastatur, Bildschirm und Diskettenlaufwerken zum intelligenten Peripherie-Controller für den SBC 8086.

Bürocomputer-Consolprogramm und SBC-Urlader

Das Consol-Programm CONS.COM wird unter dem Betriebssystem SCPX 1526 angewendet und benutzt für die Verwaltung von Tastatur, Bildschirm und Laufwerken die BDOS-Funktionen, so daß die Bedienung des SBC 8086 an allen unter SCPX 1526 arbeitenden Rechnern möglich ist. In der gegenwärtigen Generierungsvariante erfolgt der Datentransfer über eine serielle IFSS-Schnittstelle.
Der EPROM-residente SBC-Urlader führt nach RESET folgende Operationen aus:

– Grundinitialisierung des Systems
– Programmierung der Terminal-Ports
– einfacher RAM-Test von Daten- und Stacksegment
– Bildschirmausgabe eines Bedienermenüs.

Im Menü sind folgende Funktionen wählbar:
– RAM-Test mit den Teilfunktionen: RAM-Suche und vollständiger RAM-Test
– Load: Laden einer .H86 Datei von der Diskette in den RAM-Speicher des SBC 8086
– Go: Start eines mit load geladenen Anwenderprogramms
– Monitor-Start: Start des mit load geladenen Monitors
– CPM86 Load&Go: Laden und Start des 16-Bit-Betriebssystems SCP 1700 (kompatibel CP/M 86) /6/.

16-Bit-Monitor MON86

Der Monitor MON86.H86 mit 16 KByte wird mit der Load-Funktion in den SBC-RAM geladen (Adreßbereich 1C000 ... 1 FFFF). Das Kernstück dieses Monitors ist der 957B-Debugger /4/ mit folgenden Grundfunktionen:
– Go: Start eines Anwenderprogramms ab Startadresse mit max. 4 Breakpoints
– Single-Step: Einzelschrittbetrieb unter Flag-Steuerung
– Display: Anzeige eines Speicherbereiches
– Substitute: Veränderung des Inhaltes einzelner Speicheradressen
– Move: Transfer eines Speicherbereiches zu einem anderen
– Compare: Vergleich zweier Speicherbereiche
– Find: Suchen einer Datenfolge bis max. 16 Bytes
– Reassembler: Reassemblieren eines Maschinenprogramms mit mnemotechnischer Darstellung
– Examine: Anzeige und Veränderung der CPU-Register
– Port-Input: Eingabe von einem Port
– Port-Output: Ausgabe zu einem Port.

Der Monitor MON86 entstand aus dem 957B-Debugger, einigen Funktionen des Urladers
– Load
– RAM-Test
– CPM Load&Go

und folgender Zusatzfunktion:
– Save: Abspeichern eines SBC-RAM-Bereiches als .H86-Datei auf Diskette.

Applikative Erfahrungen mit dem SBC 8086

Der im Beitrag vorgestellte Single-Board-Computer SBC 8086 wurde im Wissenschaftsbereich Mikrorechentechnik/Schaltungstechnik der Wilhelm-Pieck-Universität Rostock entwickelt und stellt in der oben beschriebenen Konfiguration ein leistungsfähiges 16-Bit-Mikrorechnersystem dar. In dieser Einheit wird der SBC 8086, beginnend mit dem Jahr 1988, im Wissenschaftlichen Gerätebau der Technischen Universität Dresden (ZWGB) in einer Kleinserie produziert. Die Erfahrungen mit dem Single-User-Betriebssystem, kompatibel zum SCP 1700, und die sich daraus ergebenden Schlußfolgerungen sind in dem anschließenden Beitrag /7/ beschrieben. Weiterhin wurden applikative Untersuchungen zur Multiprozessorfähigkeit des Systems 8086 durchgeführt. Die implementierten Mechanismen zum Multiprocessing erlauben die Ergänzung der 8086-CPU mit ihren Coprozessoren 8087-Arithmetik-Prozessor und 8089-I/O-Prozessor.
Eine Sandwich-Anordnung vereinigt diese 3 Prozessoren auf einer in die CPU-Fassung steckbaren Einheit zu einem Multiprozessorsystem. Der Arithmetik-Prozessor stellt für den Anwender eine transparente Erweiterung des Register- und Befehlssatzes der Master-CPU dar, wobei eine erhebliche Verkürzung der Ausführungszeiten und Vergrößerung der Verarbeitungsbreite erreicht wird. Der I/O-Prozessor 8089 vereinigt einen DMA-Modul mit einer auf Peripheriesteuerung spezialisierten CPU und entlastet in einem 8086-System die Master-CPU von zeitaufwendigen E/A-Operationen.
Zusammenfassend kann festgestellt werden, daß der SBC 8086 eine ökonomische Lösung zum Einstieg in die 16-Bit-Mikrorechentechnik darstellt. Die Firmware für Entwicklung und Test ermöglicht eine effiziente Programmentwicklung. Zusätzlich kann der SBC 8086 als OEM-Rechner für die Prozeßautomatisierung eingesetzt werden.

Literatur

/1/ Single Board Computer Hardware Reference Manual. iSBC 86/12A, INTEL
/2/ Arbeitsplatzcomputer A 7100. edv-aspekte 6 (1987) 1
/3/ Microsystems Components Handbook, INTEL 1986
/4/ User's Guide for the iSBC 957B iAPX 86,88 Interface and Execution Package. INTEL
/5/ Dorfmüller, L.: Despang, H.-G.: Entwicklungsunterstützung für 16-Bit-Mehrmikrorechnersysteme. Mikroprozessortechnik, Berlin 1 (1987) 2, S. 51
/6/ Anwenderdokumentation SCP 1700. VEB Kombinat Robotron
/7/ Kabatzke, W.: Single-User-Betriebssystem für den SBC 8086. Mikroprozessortechnik, Berlin 1 (1987) 7, S. 203

⊠ KONTAKT ☎

Wilhelm-Pieck-Universität, Sektion Technische Elektronik, Wissenschaftsbereich Mikrorechentechnik/Schaltungstechnik, Albert-Einstein-Str. 2a, Rostock, 2500; Tel.: 45366

Single-User-Betriebssystem für den SBC 8086

Wolfgang Kabatzke
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik

Der Artikel beschreibt, wie mit Hilfe eines 8-Bit-Bürocomputers und eines 8086-kompatiblen 16-Bit-Mikrorechners die Implementierung eines Single-User-Betriebssystems erfolgen kann. Der 16-Bit-Mikrorechner stellt eine Hardwareapplikation in Form eines SBC oder Modulrechners dar, jedoch ohne Massenspeicher, Bildschirm und Tastatur. Die Implementierung nutzt Softwareanteile des 8-Bit-Bürocomputers und ermöglicht eine Nutzung der verfügbaren Dienstprogramme des Betriebssystems (kompatibel zu CP/M-86).

Einführung

Viele Anwendungsaufgaben der Mikrorechentechnik, die hohe Anforderungen an das Echtzeitverhalten der Mikrorechnersysteme bezüglich der Rechen- und Verarbeitungszeiten stellen (z. B. spezielle Roboter- und Automatisierungsanlagensteuerungen, Datenbanksysteme), lassen sich nicht mehr durch Einprozessor-/Einrechnerlösungen auf der Basis der 8-Bit-Mikrorechentechnik lösen.
Als mögliche Auswege zur Lösung dieses Problems bieten sich drei alternative Lösungswege an:
① Schaffung eines Mehrrechner-/Mehrprozessorsystems auf der Grundlage der 8-Bit-Mikrorechentechnik als Mehrrechner-/Mehrprozessormastersystem
② Konsequenter Einsatz eines 16-Bit-Mikrorechnersystems
③ Einsatz einer Mischkonfiguration aus einem 16-Bit-Mikrorechner und einem oder mehreren 8-Bit-Mikrorechnern, wobei der 16-Bit-Mikrorechner als Masterrechner fungiert.

Bewertung dieser Varianten:

Zu ①:
Ein Mehrrechner-/Mehrprozessormastersystem wirft immer die Frage nach der Programmierung eines derartigen Systems auf, da im Hinblick auf eine universelle Softwarenutzung, vor allem der Applikationsprogramme, standardisierte Betriebssysteme verwendet werden sollten. Betriebssysteme, die auf einer derartigen Anordnung laufen, sind derzeit nicht verfügbar, so daß als einzige Alternative die Selbsterstellung eines Betriebssystems notwendig wäre. Der Aufwand hierfür bindet jedoch erhebliche Kapazitäten und ist daher nicht vertretbar.

Zu ②:
Der Einsatz eines 16-Bit-Mikrorechnersystems stellt eine effektive Lösung der Problematik dar, da durch die Befehlsstruktur, durch das Befehlspipelining und die Interruptstruktur die Lösung von hochwertigen Anwendungsaufgaben der Mikrorechentechnik möglich wird.
Standardisierte Betriebssysteme sind für derartige Systeme verfügbar, zum Beispiel für den 16-Bit-Prozessor K1810 WM86 (kompatibel zu 8086) die Betriebssysteme SCP 1700 (kompatibel zu CP/M-86) und BOS

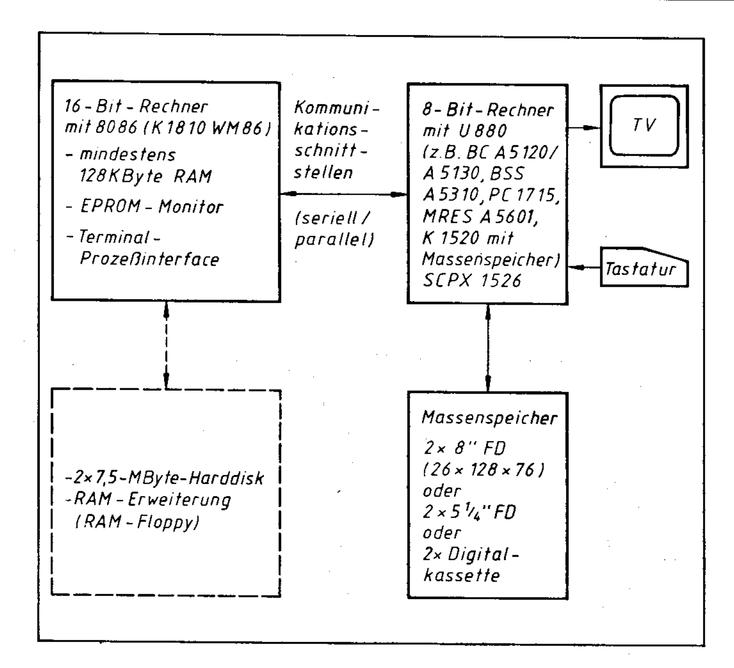


Bild 1 Gerätetechnisches Konzept

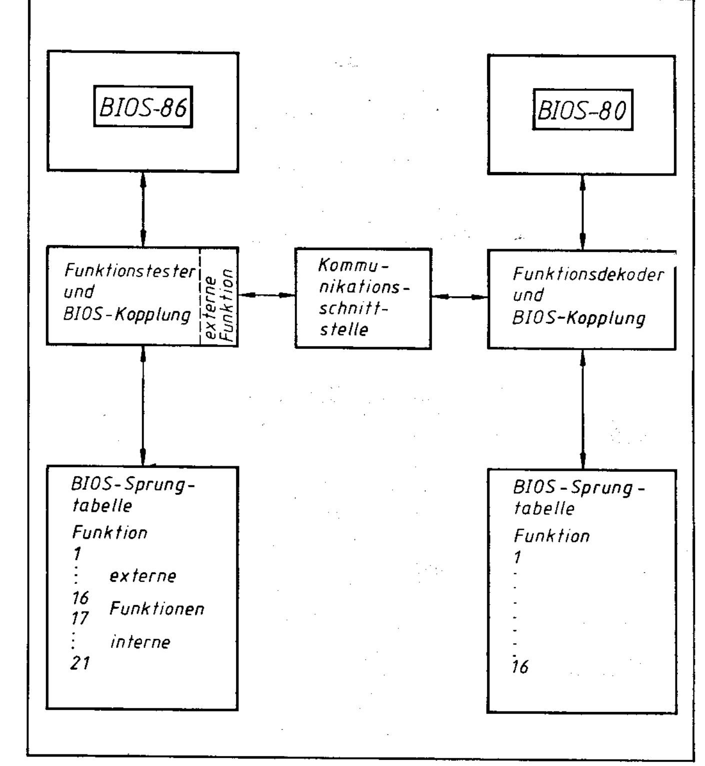


Bild 2 BIOS-Verkopplung

1810 (kompatibel zu iRMX 86). Diese Betriebssysteme erfordern in ihrer Ursprungsversion jedoch eine spezielle Hardwarestruktur (Speicher > 128 KByte, E/A-Controller), die nicht in jedem Fall verfügbar ist.
Einen Ausweg bietet der Selbstbau solcher Systeme, jedoch sind hier Grenzen gesetzt durch die praktische Realisierbarkeit der Baugruppen (Mehrebenentechnik > 2 im Laborbetrieb technologisch nicht beherrschbar).

Zu ③:

Diese Variante stellt die Kombination eines peripherielosen 16-Bit-Mikrorechners und eines oder mehrerer 8-Bit-„Peripheriemikrorechner" dar, das heißt, die Ansteuerung der peripheren Geräte wird vom 8-Bit-„Peripheriemikrorechner" übernommen, der über serielle/parallele Kommunikationsschnittstellen an den 16-Bit-Mikrorechner angekoppelt ist.
Diese Variante stellt für die gegebene Situation und auch aus Kostengründen solange die optimale Lösung dar, bis reine 16-Bit-Systeme verfügbar sind. Sie erlaubt uns, mit der 16-Bit-Technik arbeiten zu können und sie dabei näher kennenzulernen.

Gerätetechnisches Konzept

Die konzipierte Gerätekonfiguration ist in Bild 1 dargestellt. Der verwendete 8-Bit-Mikrorechner auf der Basis des Prozessors U 880 muß über Massenspeicher verfügen (Kassetten- oder Diskettenlaufwerke), wobei im Sinne eines schnellen Massenspeicherzugriffs den Diskettenlaufwerken der Vorzug gegeben werden sollte.
Es lassen sich folgende 8-Bit-Mikrorechner einsetzen:

- BC A 5120
- BC A 5130
- BSS A 5310
- Terminals K 892X
- PC 1715
- Eigenbau-BC auf der Basis des K-1520-Systems (z. B. umgerüstetes Terminal K 8912, K 8913).

Diese Rechner sind mit der erforderlichen Peripherie (Tastatur, Bildschirm und mindestens 2 Kassetten-/Diskettenlaufwerke) ausgerüstet und verfügen über serielle Schnittstellen (V.24/IFSS) zur Ankopplung an den 16-Bit-Mikrorechner. Mit geringem Aufwand sind parallele Interfaces nachrüstbar.
Der verwendete 16-Bit-Mikrorechner auf der Basis des Prozessors 8086 muß folgende Baugruppen enthalten:

- CPU mit Clockgenerator 8284 und Buscontroller 8288
- mindestens 128 KByte RAM
- mindestens 4 KByte ROM/EPROM zur Ablage eines einfachen Laders/Monitors
- Interruptsteuerlogik mit PIC 8259 A
- Peripherieschaltkreise (USART 8251 A, TIMER 8253 (8253 A), PPI 8255 A zur Terminal- und Prozeßkopplung.

Diese Minimalversion wurde im Wissenschaftsbereich Mikrorechentechnik/Schaltungstechnik in unserer Sektion als *Single-Board-Computer 8086* (SBC 8086) realisiert und wird in Forschung und der Lehre eingesetzt /1/.
Als Vorläufer des SBC wurde eine modular gestaltete Geräteversion in Form des *Applikationsrechners 8086* in unserem Wissenschaftsbereich entwickelt und aufgebaut. Dieser enthält neben o. g. Baugruppen einen Speicher von 786 KByte DRAM, einen komfortablen EPROM-Monitor mit Debugger und Reassembler und, zusätzlich zum Interface, spezielle Prozeß-E/A-Baugruppen auf der Basis der Interface-Schaltkreise PIO, SIO, CTC und DMA des Systems U 880. Der 8086-Rechner ist über ein lokales Netz (LAN) mit weiteren 8-Bit-Bürocomputern verbunden. Der Anschluß zweier 7,25-MByte-Harddisk (EC 5052) ist realisiert. Dieser Rechner ist speziell zur Realisierung von Entwicklungssystemen und zur Echtzeitsteuerung für die Prozeßautomatisierung auf der Basis des 8086 geschaffen worden.
Beide Gerätekonzepte sind aufwärtskompatibel, das heißt, der SBC 8086 ist hardwaremäßig im Applikationsrechner enthalten, und widerspiegeln somit ein einheitliches Grundkonzept mit unterschiedlichem Ausbaugrad.

Realisierung des Single-User-Betriebssystems

Zur Generierung des Betriebssystems auf der vorgestellten Gerätekonfiguration waren folgende Softwarewerkzeuge erforderlich:
- ein lauffähiges SCPX 1526 mit mindestens 2 Diskettenlaufwerken (bis zu 4 sind möglich; für Kassetten EMOS 1520 (HfOe))
- die Disketten-/Kassettenparametertabellen des jeweils verwendeten 8-Bit-Betriebssystems
- die komplette Dokumentation zur Installierung des 16-Bit-Betriebssystems auf dem 8086-Rechner, insbesondere die BIOS-Generiervorschrift (siehe /2/ ... /5/).

Aufgrund der Aufwärtskompatibilität des 16-Bit- zum 8-Bit-Betriebssystem bezüglich des BIOS-Sprungvektors, der Parameterübergabe in bestimmten CPU-Registern und der logischen Sektorlänge (zur Datenverwaltung auf dem Massenspeicher) von 128 Byte, konnte auf BIOS-Bestandteile des 8-Bit-Systems über ein Vermittlungsprogramm direkt zurückgegriffen werden. Dadurch reduzierte sich der Aufwand für die Erstellung eines lauffähigen BIOS-86 auf ein Minimum.
Im BIOS-86 wurden allen BIOS-Funktionen Funktionsnummern vergeben (siehe auch /2/). Einige BIOS-Funktionen können dabei innerhalb des BIOS-86 ohne Kommunikation mit dem BIOS-80 ablaufen (z. B. SETDMA, GETIOBF, SETIOBF, SETDMAB, GETSEGT).
Ist eine Kommunikation mit dem BIOS-80 erforderlich, werden diesem die Funktionsnummer und die eventuell erforderlichen Parameter und Daten (z. B. bei READ und WRITE) übermittelt. Das BIOS-80 selektiert die Funktionsnummer und führt die geforderte Funk-

tion aus. Eventuell zu retournierende Parameter oder Daten werden übertragen. Äußerst wichtig ist hierbei, daß die Disketten-/Kassettenparametertabellen des BIOS-80 mit in das BIOS-86 implementiert werden, da diese dem BIOS-86 sonst nicht zugänglich sind. Ebenso muß in das BIOS-86 ein eigener DMA-Puffer und Directory-Puffer implementiert werden, da das BIOS-80 und das BIOS-86 in physikalisch getrennten Speichern liegen.
Das Wirkprinzip der BIOS-Verkopplung zeigt Bild 2. Insgesamt wurden 3 Softwarekomponenten realisiert:
1. ein Konsolenprogramm, da der Bürocomputer während des Monitorbetriebes des 8086-Rechners als Konsole arbeitet (Gleichzeitig enthält dieses Programm den bürocomputerseitigen Anteil des Bootstrapladers des 16-Bit-Betriebssystems.)
2. ein spezielles BIOS-86, welches mit dem CCP und BDOS das komplette 16-Bit-Betriebssystem bildet
3. ein BIOS-Adaptionsprogramm, welches im Bürocomputer die Kopplung zwischen dem speziellen BIOS-86 und dem BIOS-80 realisiert.

Erfahrungen mit dem 16-Bit-Single-User-Betriebssystem

Erste Erfahrungen zeigen, daß das 16-Bit-System in der beschriebenen Version lauffähig ist. Folgende Anteile des SCP 1700 /6/ können auf diesem System bereits genutzt werden:

– DDT 86	– LINK 86
– SID 86	– GENCMD
– ASM 86	– GENDEF
– RASM 86	– SUBMIT
– LIB 86	– HELP
– XREF 86	– TOD
– ED	– PRINTER
– PIP	– WS
– TURBO-PASCAL 3.01	– STAT
– DBASE III 2.4	– MBASIC 86
– FIGFORTH 86	– C-Compiler

Erste eigene Test- und Applikationsprogramme wurden erstellt und werden genutzt.
Ein Problem bildet die Datenübertragung zwischen dem Bürocomputer und dem 8086-Rechner, wenn sie seriell (V.24/IFSS) mit einer effektiven Datenübertragungsrate von 4,8 kBaud erfolgt. Dieses Verfahren führt bei längeren zu ladenden Programmen zu relativ langen Programmladezeiten (z. B. DDT86 in 40 Sekunden, Länge des DDT86: 16,5 KByte).
In der gegenwärtigen Phase wird der Systemanlauf wie folgt vollzogen:

1. Reset 8086-Rechner
2. Reset Bürocomputer und Start SCPX 1526
3. Start „Konsole-8086" auf dem Bürocomputer
4. Start „Monitor-8086" auf dem 8086-Rechner
5. Aufruf „Bootstraplader"
6. Automatisches Laden des 16-Bit-Betriebssystems vom Bürocomputer in den 8086-Rechner
7. Übergang des Bürocomputers vom Konsolenmodus in den BIOS-Modus
8. Synchronisation beider Rechner und Start des 16-Bit-Betriebssystems.

Dipl.-Ing. Wolfgang Kabatzke (29) studierte von 1978–1983 an der Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, in der Fachrichtung Technische Kybernetik und Automatisierungstechnik. Seit 1984 ist er Assistent im Wissenschaftsbereich Mikrorechentechnik/Schaltungstechnik der WPU und befaßt sich seit 1986 mit Arbeiten zur effektiven Betriebssystem-Softwaregestaltung für inhomogene Mehrrechnersysteme auf der Basis der Prozessoren 8086 und U880 unter Nutzung von Hochsprachen.

Das Laden und Starten des 16-Bit-Systems über V.24/IFSS dauert etwa 45 Sekunden.

Vervollkommnung des Betriebssystems

Folgende Arbeiten zielen auf eine Verbesserung der Leistungsparameter und stehen kurz vor ihrer Realisierung:

1. Übergang zur Rechnerkopplung über ein paralleles Interface (kompatibel SCSI-Interface) zwischen dem Bürocomputer und dem 8086-Rechner

2. Anschluß zweier 7,25-MByte-Harddisk EC 5052 an den 8086-Rechner zur Auslagerung von Systemanteilen und Applikationsprogrammen. Die Harddisk wird eine entscheidende Zugriffsverkürzung zum Massenspeicher erbringen. Die Diskette soll nur noch zum Daten- und Programmaustausch genutzt werden (Bild 1, gestrichelt umrandet). Das Laden des Betriebssystems von der Harddisk dauert etwa 3,0 Sekunden (Länge etwa 21 KByte).

3. Implementation von 2 RAM-Floppies im 8086-Applikationsrechner ebenfalls zur effektiven Massenspeicher-Zugriffsgestaltung (Bild 1, gestrichelt umrandet).

4. Nutzbarmachung der Programmiersprachen C und PASCAL und Nutzung eines Echtzeitbetriebssystems (in C geschrieben) für Prozeßanwendungen.
In einem später zu veröffentlichenden Artikel wird auf die Ergänzungen und Erweiterungen näher eingegangen. Interessenten können die vorliegenden Ergebnisse nachnutzen.

Literatur

/1/ Münzer, B.-G.; Stachowiak, T.: 16-Bit-Single-Board-Computer SBC 8086. Mikroprozessortechnik 1 (1987) 7, S. 200
/2/ CP/M-86 System Guide. Digital Research 1981
/3/ CP/M-86 Release Notes. Digital Research 1981
/4/ CP/M-86 Operating System Command Summary. Digital Research 1981
/5/ Dahmke, M.: The Byte Guide to CP/M-86. Mc Graw-Hill Book Company New York 1984
/6/ Dokumentation SCP 1700. C101X – 0000 – 1 M3030. VEB Robotron-Elektronik-Dresden
/7/ CP/M-86 Programmer's Guide. Digital Research 1981
/8/ CP/M-86 User's Guide. Digital Research 1981
/9/ Junge, S.; Keller, D.: Das Mikrorechnermodulsystem 16 und sein Einsatz im Arbeitsplatzcomputer robotron A 7100. Neue Technik im Büro, Berlin 29 (1985) 3, S. 81–87
/10/ Schoenwald, U.: Realisierung von Software- und Hardwarekomponenten für einen 16-Bit-Applikationsrechner.
Diplomarbeit an der WPU Rostock, WB AS 1985

⊠ KONTAKT ☏

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, Wissenschaftsbereich Mikrorechentechnik/Schaltungstechnik, Albert-Einstein-Straße 2, Rostock 6, 2500; Tel. 45387

Termine

Kolloquien des Informatik-Zentrums an der Technischen Universität Dresden für das 2. Halbjahr 1987

Computergrafik – Gerätetechnik, Software, Perspektiven
23. September 1987
Leitung: Doz. Dr. sc. techn. Schnabel
Wissenschaftsbereich Systemsoftware

Internationaler Stand und Entwicklungstendenzen auf dem Gebiet der Klein- und Mikrorechentechnik
29. September 1987
Leitung: Doz. Dr. sc. techn. Horn
Bereich Applikation und Service

Anforderungen an die Parallel- und Echtzeitprogrammierung
30. September 1987, 13.15 Uhr
Leitung: Prof. Dr. rer. nat. habil. Stiller
Wissenschaftsbereich Theoretische Informatik

Hard- und Softwarekomponenten zum Aufbau von lokalen Rechnernetzen
5. November 1987, 13.00 Uhr
Leitung: Prof. Dr. sc. techn. Löffler
Wissenschaftsbereich Rechnersysteme

Moderne Betriebssysteme
25. November 1987
Leitung: Prof. Dr. Kalfa
Wissenschaftsbereich Systemsoftware

Hard- und Softwarelösung für Steueraufgaben (-funktionen) in einer Fließlinie der Möbelindustrie auf der Basis Einchip-Mikrorechner als Steuerkern
8. Dezember 1987, 13.30 Uhr
Leitung: Prof. Dr. sc. techn. Meinhardt
Wissenschaftsbereich Grundlagen der Informatik

Teilnahmemeldungen (mit Angabe des Wissenschaftsbereichs) werden erbeten an das Informatik-Zentrum des Hochschulwesens der DDR an der TU Dresden, Mommsenstr. 13, Dresden, 8027.

Wechselplattencontroller für 8- und 16-Bit-Mikrorechner

Klaus Graumann, Klaus Kopplow
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik

1. Vorbemerkungen

Leistungsfähige 16-Bit-Mikrorechnerbetriebssysteme benötigen neben einem großen Hauptspeicher auch einen großen schnellen Hintergrundspeicher. International werden üblicherweise Winchesterlaufwerke mit Speicherkapazitäten von 10 ... 30 MByte verwendet. Im folgenden wird eine Lösung beschrieben, die Wechselplattenspeicher vom Typ EC 5052 als schnellen Hintergrundspeicher für 8- und 16-Bit-Mikrorechner nutzbar macht. Diese Wechselplattenspeicher werden zunehmend bei der Modernisierung von Rechenzentren freigesetzt, sind aber für den Einsatz in der Lehre und Ausbildung meistens noch geeignet. Der Wechselplattenspeicher EC 5052 besitzt ein Bruttospeichervolumen von 7,25 MByte, arbeitet mit einer Übertragungsgeschwindigkeit von 1,25 MBit/s und verwendet zur Aufzeichnung der Daten das FM-Verfahren /1/.

Als Zielsystem für den Anschluß des EC 5052 diente ein Bürocomputer A 5120.16 mit dem Betriebssystem MUTOS 8000. Des weiteren können beliebige 8- und 16-Bit-Mikrorechnersysteme leicht angepaßt werden. Deshalb wurde der Anschluß des Wechselplattenspeichers als intelligente, eigenständige und abgesetzte Einheit konzipiert – als Controller (Bild 1). Der Wechselplattencontroller beinhaltet folgende Baugruppen:

- ZRE K 2521
- RAM-Speicher, z. B. OPS K 3520
- Interfacebaugruppe für den WPS 5052
- Parallel-Serien-Wandler
- Interfacebaugruppe Host.

Der Anschluß des Controllers erfolgt über ein paralleles Interface an den Bus des jeweiligen Host-Mikrorechners. Der Aufbau des Controllers wurde modular gewählt, um eine einfache Anpassung an andere Massenspeicher und andere Mikrorechner vornehmen zu können.

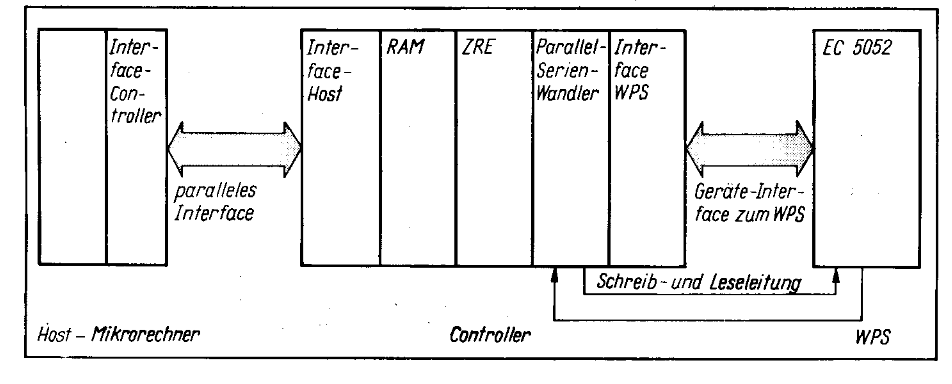


Bild 1 Konfiguration des WPS-Anschlusses an 8- oder 16-Bit-Mikrorechner

2. Hardwarekomponenten

Die *Interfacebaugruppen für den WPS EC 5052* beinhaltet gepufferte E/A-Tore und Leitungstreiber bzw. Leitungsempfänger für alle Steuer- und Statussignale des WPS sowie für den Gerätebus einwärts und den Gerätebus auswärts.

Die Baugruppe *Parallel-Serien-Wandler* hat die Aufgabe, die 8-Bit-parallelen Daten zu serialisieren, zu kodieren, an die Leitungen anzupassen und eine Blocksicherung mittels CRC vorzunehmen. Die gelesene Information wird vom Parallel-Serien-Wandler dekodiert, aus dem FM-Takt-Daten-Gemisch werden Takt und Daten separiert. Es erfolgen eine Prüfung des CRC-Blocksicherungszeichens und die Wandlung der seriellen Daten in 8-Bit-parallele Daten. Die Aufzeichnung der Daten auf den Datenträger erfolgt hardsektoriert mit 512-Byte-Blöcken. Dadurch entsprechen die logischen Blöcke des MUTOS 8000 den physischen Blöcken auf der Wechselplatte. Kernstück der Baugruppe Parallel-Serien-Wandler ist ein SIO-Baustein U 8560.

Für die direkte Aufzeichnung von Daten mit 1,25 MBit/s auf die Wechselplatte ist der SIO-Baustein jedoch nicht geeignet. Zu diesem Zweck wurde je ein serieller FIFO mit der Tiefe eines Datenblockes von 512 Byte vorgesehen. Die Schreibdaten werden zuerst mit dem SIO entsprechend serialisiert und mit Blocksicherungszeichen versehen in den seriellen Schreibpuffer geschrieben. Dieses erfolgt mit einer Datenrate von 500 KBit/s. Aus dem Schreibpuffer gelangen dann die Schreibdaten mit einer Datenrate von 1,25 MBit/s über den Encoder auf die Wechselplatte. Die maximal erreichbare Datenrate wird prinzipiell nur durch die Speicherzugriffszeit begrenzt. Für die Puffer wurden Speicher 4 K × 1 Bit (KR 132 RU 5) eingesetzt. Es wären, je nach eingesetztem Speichertyp, weitaus größere Datenraten erreichbar.

(Fortsetzung auf Seite 212)

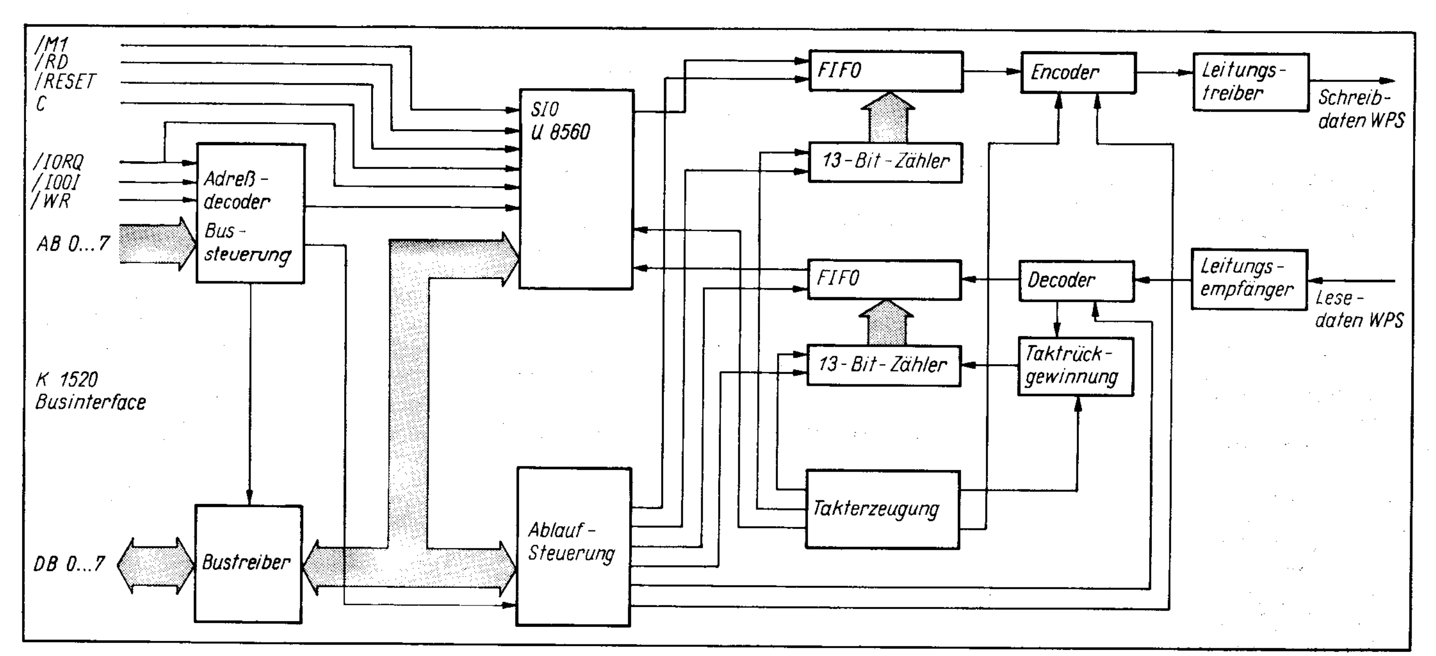


Bild 2 Struktur der Baugruppe Parallel-Serien-Wandler

Programmieren mit MACRO-SM

Teil I

Dr. Thomas Horn
Informatikzentrum des Hochschulwesens
an der Technischen Universität
Dresden

Die Makroassemblersprache MACRO-SM ist eine moderne flexible Programmiersprache zur Programmierung von Klein- und Mikrorechnern des Systems der Kleinrechentechnik (SKR) der Länder des Rates für Gegenseitige Wirtschaftshilfe (RGW). Während auf den traditionellen Kleinrechenanlagen des SKR die höheren Programmiersprachen wie FORTRAN-77, C, COBOL und PASCAL eine wachsende Bedeutung erfahren haben, hat die Makroassemblersprache MACRO-SM in den letzten Jahren vor allem mit den neuen sowjetischen Mikroprozessorschaltkreisreihen KR 588 und K 1801 für den Einsatz in vielen Geräten und Anlagen der Automatisierungstechnik an Bedeutung gewonnen. Die genannten Mikroprozessorsysteme sind zu den Rechenanlagen SM 4 bzw. Elektronika-60 (M) voll befehlskompatibel.
Mit dieser Artikelreihe soll deshalb der gewachsenen Bedeutung der SKR-Technik Rechnung getragen und dem Leser eine Einführung in die maschinenorientierte Programmierung von Rechenanlagen des SKR und befehlskompatibler Mikroprozessorsysteme gegeben werden.

0. Einleitung

Die Grundmodelle des SKR bilden die Kleinrechnersysteme SM3 und SM4. Sie stellen leistungsfähige Kleinrechnersysteme mit Busstruktur und gleichem Grundbefehlssatz (BIS – Basic Instruction Set) dar, die sowohl für den wissenschaftlich-technischen und ökonomischen Einsatz als auch für den Prozeßrechnereinsatz im Rahmen der Labor- und Prozeßautomatisierung vorgesehen sind. Eine Übersicht über wichtige SKR-Anlagen, die in der DDR im Einsatz sind, ist in Tafel 1 enthalten.
Die Modelle vom Typ SM4 zeichnen sich gegenüber dem Modell SM3 durch eine größere Hauptspeicherkapazität, einen erweiterten Befehlssatz (EIS – Extended Instruction Set) und eine Unterstützung für die Verarbeitung von Gleitkommazahlen (FIS – Floating Instruction Set oder FPP – Floating Point Processor) aus. Bezüglich der Ein- und Ausgabemoduln besteht zwischen allen SKR-Anlagen eine volle Kompatibilität.
Die Kleinrechnersysteme des SKR haben eine modulare Struktur, die flexibel an den jeweiligen Einsatzfall anpaßbar ist. Die Flexibilität begründet sich vor allem auf die Anwendung eines universellen Einheitsbusses mit asynchroner Übertragung und der Möglichkeit des Anschlusses eines breiten Spektrums peripherer Geräte.
Die Kleinrechnersysteme des SKR sind gegenwärtig durch modernere Anlagen auf der Basis von hochintegrierten Schaltkreisen und einer 22-Bit-Hauptspeicherverwaltungseinheit zur Adressierung von max. 4 MByte physischer Hauptspeicherkapazität ergänzt worden.
Darüber hinaus werden die hochintegrierten Schaltkreisfamilien und die genannten Mikroprozessorsysteme verstärkt in der Labor- und Prozeßautomatisierung eingesetzt, wobei vor allem die Makroassemblerprogrammierung Anwendung findet.
In diesem MP-Kurs wird aus den genannten Gründen zuerst eine Einführung in Assemblersprache und Makrotechnik gegeben, und im letzten Teil wird der Maschinenbefehlssatz der SKR-Anlagen behandelt. Die Anwendung wird an einem abschließenden Beispiel demonstriert.

Tafel 1 Übersicht über ausgewählte Rechenanlagen des SKR

Rechenanlage	SM3–10	SM4–10	SM4–20	I100	K1630	I102F	SM14–20	SM52–11 PLUS[6]
Herstellerland	**UdSSR**	**UdSSR**	**ČSSR**	**SRR**	**DDR**	**SRR**	**UdSSR**	**ČSSR**
Einführungsjahr	1977	1978	1981	1981	1982	1983	1984	1985
max. HS-Größe [KByte]	56	248	248	248	248	248 3840[2]	248 3840[3]	248 3840[4]
Verarbeitungstyp	**Parallel, 1 Wort (16 Bit), 1 Byte (8 Bit)**							
Befehlslänge	1, 2 und 3 Worte							
Registeranzahl	8	8	8 + 6	8	8(+6)	8 + 6	8 + 6	8 + 6
Befehlssatz	BIS	BIS EIS FIS	BIS EIS FPP	BIS EIS FIS	BIS EIS (FPP)[1]	BIS EIS FPP	BIS EIS FPP	BIS EIS FPP[5]
Befehlsausführungszeit für Reg.-Reg.-Befehle [μs]	5,0	1,2	2,3	2,3	3,5–4,2	0,5	1,0	0,34

[1] Wahlweise mit FPP; [2] Wahlweise mit 22-Bit-Hauptspeicherverwaltung; [3] Seit 1985 mit 22-Bit-Hauptspeicherverwaltung; [4] Seit 1986 mit 22-Bit-Hauptspeicherverwaltung; [5] FPP-Befehle zusätzlich auch durch das Mikroprogramm realisiert; [6] Wahlweise mit freier Mikroprogrammierung

1. Einführung in die Makroassemblersprache

MACRO-SM ist eine moderne Assemblersprache, die sich durch Flexibilität, Transparenz und ein entwickeltes Makrokonzept auszeichnet. Die Übersetzung der Programme erfolgt im allgemeinen in eine Objektsprache. Ein Programm in der Objektsprache wird als Objektmodul bezeichnet, ist verschieblich und kann vom Taskbuilder in eine absolute, nicht verschiebliche, abarbeitungsfähige Task (Lademodul) umgewandelt werden. Der Taskbuilder unterstützt dabei die Verbindung einer Vielzahl von Objektmoduln zu einer Task, wobei infolge der Realisierung globaler Symbole in der Makroassemblersprache zwischen den Objektmoduln Querbezüge hergestellt werden können. Durch diese Eigenschaften wird insbesondere eine modulare Programmierung unterstützt, die eine bessere Ausnutzung der Ressourcen gestattet sowie die Transparenz der Programmsysteme und die Wiederverwendbarkeit und Änderungsfreundlichkeit der einzelnen Programmbausteine (Moduln) erhöht. Für spezielle Anwendungsfälle besteht die Möglichkeit der Übersetzung in einen absoluten Lademodul. Die Realisierung umfangreicher Möglichkeiten der bedingten Übersetzung gestattet die Generierung von an den jeweiligen Anwendungsfall angepaßten Programmsystemen aus einem Quellmodul. Das ausgereifte Makrokonzept stellt ein effektives Rationalisierungsmittel für die Programmentwicklung dar. Es besteht neben der Verwendung von programmspezifischen Makrodefinitionen die Möglichkeit der Anwendung einer umfangreichen Systemmakrobibliothek und nutzereigener Privatbibliotheken. Die hohe Flexibilität des Assemblers drückt sich außerdem im Vorhandensein umfangreicher Möglichkeiten zur Steuerung des Übersetzungsprozesses und der Protokollgestaltung aus.

1.1. Das Format der Anweisungen

Eine Anweisung kann im freien Format aus

maximal vier Feldern bestehen:

name: operation operand(en) ; kommentar

Eine *Anweisung* wird immer als Quellzeile betrachtet und in einer Druckzeile protokolliert. Die maximale Länge einer Zeile darf 132 Zeichen betragen. Es wird aber auf Grund von Einschränkungen bei verschiedenen E/A-Geräten empfohlen, die Zeilenlänge von 80 Zeichen nicht zu überschreiten. Jedes der vier Felder einer Anweisung kann bei der Programmierung ausgelassen werden. Wenn alle vier Felder ausgelassen werden, entsteht eine Leerzeile (Leeranweisung). Leerzeilen sind im Programm zulässig. Eine Anweisung wird vom Makroassembler von links nach rechts abgearbeitet. Wird im Namensfeld ein Symbol angegeben, so wird dem Symbol der augenblickliche Wert des Speicherplatzzuordnungszählers (SZZ) zugewiesen.
Das *Namensfeld* wird als ein solches durch den Doppelpunkt (:) erkannt, der dem Symbol folgen muß. Im Namensfeld dürfen auch mehrere Symbole definiert werden, wobei jedes Symbol durch einen Doppelpunkt abgeschlossen werden muß und jedem Symbol der gleiche Wert zugewiesen wird.

Beispiele:
```
START:
ANFANG: MOV R0,-(SP) ; RETTEN R0
A0:B0: FELD:  .BLKW 20
```

Das Operationsfeld folgt dem Namensfeld. Bei fehlendem Namensfeld beginnt die Anweisung mit dem Operationsfeld. Im Operationsfeld steht ein Symbol, das die mnemonische Bezeichnung des Maschinenbefehls, der Assembleranweisung oder des Makrorufes festlegt. Begrenzer des Operationsfeldes sind Leerzeichen (SP), Tabulator (HT) und alle Sonderzeichen, die nicht im Radix-50-Kode (Tafel 2) enthalten sind.

Beispiele:
```
ZYKL: MOV(R0)+,R1 ;LADEN VON
A(I+1) MOV R1,A
       MOV #25,R4
```

Bei ausgelassenem Operationsfeld wird die Anweisung als eine **.WORD**-Anweisung interpretiert. Ein ausgelassenes Operationsfeld wird auch angenommen, wenn das Symbol im Operationsfeld nicht definiert ist, z. B. bei einem fehlerhaften Operationskode.
Das *Operandenfeld* spezifiziert nach dem Operationsfeld die Operanden, mit denen die Operation ausgeführt werden soll. Werden mehrere Operanden angegeben, so müssen sie untereinander durch Komma (,), Tabulator (HT) oder Leerzeichen (SP) getrennt werden.

Beispiele:
```
M25:
PUT #FELD,#80.,#40 ; AUSGABE
MOV A R0
```

Das Operandenfeld endet mit dem Semikolon (;), das einen Kommentar einleitet, oder bei fehlendem Kommentar am Zeilenende.

Tafel 2 RADIX-50-KODE

Zeichen	Erstes Kode-zeichen	Zweites Kode-zeichen	Drittes Kode-zeichen
SP	000000	000000	000000
A	003100	000050	000001
B	006200	000120	000002
C	011300	000170	000003
D	014400	000240	000004
E	017500	000310	000005
F	022600	000360	000006
G	025700	000430	000007
H	031000	000500	000010
I	034100	000550	000011
J	037200	000620	000012
K	042300	000670	000013
L	045400	000740	000014
M	050500	001010	000015
N	053600	001060	000016
O	056700	001130	000017
P	062000	001200	000020
Q	065100	001250	000021
R	070200	001320	000022
S	073300	001370	000023
T	076400	001440	000024
U	101500	001510	000025
V	104600	001560	000026
W	107700	001630	000027
X	113000	001700	000030
Y	116100	001750	000031
Z	121200	002020	000032
Sputnik	124300	002070	000033
.	127400	002140	000034
unbenannt	132500	002210	000035
0	135600	002260	000036
1	140700	002330	000037
2	144000	002400	000040
3	147100	002450	000041
4	152200	002520	000042
5	155300	002570	000043
6	160400	002640	000044
7	163500	002710	000045
8	166600	002760	000046
9	171700	003030	000047

Anmerkung:
Die Umwandlung von drei ASCII-Zeichen $\alpha_1\alpha_2\alpha_3$ erfolgt nach der Formel $\alpha_1 \cdot 50^2 + \alpha_2 \cdot 50 + \alpha$.
Beispiel: ZA5 = 121200 + 50 + 43 = 121313

Das *Kommentarfeld* kann alle druckbaren Zeichen des ASCII- oder GOST-Kodes enthalten und dient der Erläuterung der Anweisungen des Programms. Es hat keinen Einfluß auf den Übersetzungsvorgang und auf den Objektmodul. Fehlen die anderen Felder der Anweisung, so kann das Kommentarfeld ab Position 1 beginnen (Kommentaranweisung).

Beispiele:
```
START: START DES HAUPTPRO-
       ;GRAMMES FUER DIE
       ;VERSUCHSAUSWERTUNG
;BENUTZTE UNTERPROGRAMME:
;EAL, KONVD, QUAD
```

Anmerkungen:
1. Zur übersichtlichen Gestaltung der Druckprotokolle können beliebig viele Leerzeichen (SP) und/oder Tabulatoren (HT) zwischen den Feldern, Operanden bzw. Symbolen und Termen eines Ausdruckes eingefügt werden.
2. Zur einheitlichen Gestaltung der Druckprotokolle wird empfohlen, das Namensfeld ab Position 1, das Operationsfeld ab Position 9, das Operandenfeld ab Position 17 und das Kommentarfeld ab Position 33 zu beginnen. Diese Positionen werden automatisch eingestellt, wenn die Felder mit einem Tabulator (HT) (zusätzlich) begrenzt werden, da ein Tabulator immer einen Sprung auf die nächste durch 8 teilbare Position plus 1 bewirkt (9, 17, 25, 33, 41 usw.). Vor dem Kommentarfeld sind eventuell bei fehlendem oder kurzem Operandenfeld zwei Tabulatoren erforderlich.

1.2. Das Prinzip der Übersetzung

Die Übersetzung von MACRO-Quellprogrammen erfolgt nach dem Prinzip eines Zwei-Pass-Assemblers. Im ersten Assemblerpass werden die internen Datenbereiche wie dynamischer Speicherbereich, Pufferbereiche und Filesteuerbereiche initialisiert. Anschließend erfolgt die Eingabe und Analyse der Kommandozeile, die die Filespezifikationen für das Quellfile, Makrobibliotheksfile, Objektfile und Listenfile sowie weitere Steuerinformationen enthält. Nach diesen Vorbereitungen erfolgt das Einlesen der Quellzeilen und ihre Analyse.
Das Hauptziel des ersten Passes besteht im Erstellen der Nutzersymboltabelle (UST) und der Makrosymboltabelle (MST). Zu diesem Zweck muß die Makrogenerierung durchgeführt und jeder Maschinenbefehl soweit übersetzt werden, daß seine Länge bestimmt werden kann (1, 2 oder 3 Worte). Durch Addieren der Befehls- und Speicherbereichslängen wird der Speicherplatzzuordnungszähler (SZZ) gebildet, der zum Aufbauen der UST benötigt wird. Gleichzeitig wird im ersten Pass die Subtiteltabelle in das Listenfile ausgegeben.
Im zweiten Pass werden die Quellzeilen noch einmal eingelesen und im Prinzip die gleichen Arbeitsschritte wie im ersten Pass ausgeführt, mit dem Ziel der vollständigen Übersetzung der Maschinenbefehle unter Verwendung der im ersten Pass aufgebauten Symboltabelle. Es werden dabei gleichzeitig der Objektmodul und das Assemblerprotokoll (Listfile) erzeugt. Nach der Übersetzung werden die Symboltabelle und gegebenenfalls die Cross-Reference-Tabelle in das Listenfile ausgegeben.

1.3. Alphabet

Das *Alphabet* von MACRO-SM umfaßt:
- die Großbuchstaben des lateinischen Alphabets von A bis Z
- die arabischen Ziffern von 0 bis 9
- Sonderzeichen . ¤ : = % # @ ( ) , ; < > + − * / & ! " ' ^ ?
- Steuerzeichen FF HT SP.

Zusammenfassend ist die Bedeutung der einzelnen Zeichen in Tafel 3 dargestellt. Außer diesen Zeichen sind in Kommentaren alle anderen druckbaren Zeichen zulässig. Die Verwendung der kleinen lateinischen Buchstaben des ASCII-Kodes bzw. der Großbuchstaben des kyrillischen Alphabets des GOST-Kodes ist bei **.ENABL LC** möglich.

1.4. Symbole

Von der programmtechnischen Realisierung her werden die Symbole in permanente Symbole, Nutzersymbole, Makrosymbole und lokale Symbole unterschieden.

Permanente Symbole dienen zur Bezeichnung der Maschinenbefehle und Assembleranweisungen. Sie sind in der Tabelle der permanenten Symbole (PST) zusammengefaßt.
Nutzersymbole sind vom Nutzer im Programm definierte Symbole, denen entweder der augenblickliche Wert des SZZ (:) oder ein beliebiger anderer Wert durch die Direktanweisung (=) zugewiesen wird. Nutzersymbole werden im ersten Assemblerpass in die Nutzersymboltabelle (UST) eingetragen. Die UST wird nach der Übersetzung gedruckt (außer bei **.NLST SYM**).
Makrosymbole sind die symbolischen Namen von Makrodefinitionen, die entweder im Programm definiert (**.MACRO**) oder aus einer Makrobibliothek aufgerufen (**.MCALL**) werden können. Sie werden ebenfalls im ersten Pass in die Makrosymboltabelle (MST) eingetragen.
Für die Bestimmung eines Symbols ist die Reihenfolge des Suchens in den Tabellen entscheidend. Die Symbole des Operationsfeldes werden wie folgt in den Tabellen gesucht: MST, PST, UST.
Lokale Symbole sind spezielle numerische Symbole (z. B. **1⊠**, **2⊠** usw.) mit einem eingeschränkten Gültigkeitsbereich, die ähnlich wie Nutzersymbole verwendet werden. Besonders vorteilhalft werden sie für interne Sprungmarken verwendet, wenn dem Sprungziel keine mnemonische Bezeichnung zugeordnet werden braucht. Lokale Symbole werden in lokale Symbolblöcke (LSB) eingetragen. Auf Grund des eingeschränkten Gültigkeitsbereichs können automatisch oder nutzergesteuert mehrere LSB nacheinander eingerichtet werden. Ein lokales Symbol belegt nur 3 Worte im LSB, während ein Symbol in der UST 4 Worte belegt.

1.4.1. Nutzer- und Makrosymbole

Nutzer- und *Makrosymbole*, wie auch permanente Symbole, werden nach gleichen syntaktischen Regeln gebildet. Da diese Symbole intern im Assembler im Radix-50-Kode dargestellt werden, sind für die Bildung von Symbolen alle Radix-50-Zeichen zulässig:

- Großbuchstaben des lateinischen Alphabets
- arabische Ziffern
- Sonderzeichen **.** und ⊠

Die Sonderzeichen **.** und ⊠ werden als Alphazeichen gewertet und sind für die Bildung von Systemsymbolen reserviert. Zur Vermeidung von Konflikten sollten diese Zeichen vom Anwendungsprogrammierer nicht benutzt werden.
Ein Symbol besteht aus 1 bis 6 Zeichen, wobei das erste Zeichen immer ein Alphazeichen sein muß. Längere Symbole sind zulässig, es sind aber nur die ersten 6 Zeichen signifikant.

Beispiele:
A, R0, A250WZ, ALPHA, ALPHA250, ALPHA251
.WRITE, ⊠⊠⊠SYM, .WORD, ⊠.25., A⊠DEZ

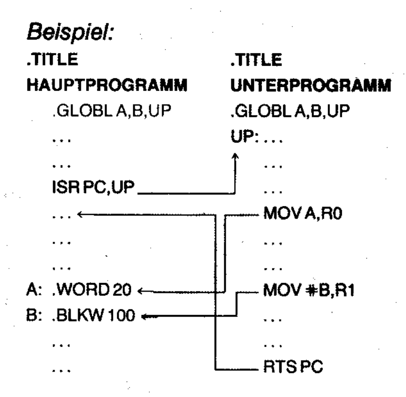
Die Symbole ALPHA250 und ALPHA251 sind somit identisch und entsprechen dem Symbol AIPHA2.
Hinweis:
Ein Symbol wird durch Leerzeichen (SP) und jedes nicht im Radix-50-Kode definierte Symbol begrenzt.
Der Gültigkeitsbereich der Nutzer- und Makrosymbole erstreckt sich über das gesamte Programm, unabhängig von einer Aufteilung in Programmabschnitte. Diese Symbole werden deshalb als interne Symbole eines Programms (einer Übersetzungseinheit) bezeichnet.
Die Herstellung von Querbezügen zwischen getrennt übersetzten Quellprogrammen ist mit dem Taskbuilder unter Verwendung globaler Symbole möglich. Man unterscheidet dabei zwischen *globalen Definitionen* und *globalen Bezugnahmen*. Ein Symbol ist global definiert, wenn es im Programm *global-definiert* ist (:: bzw. ==) oder durch eine **.GLOBL**-Anweisung als globales spezifiziert wird.

Tafel 3 Alphabet der Assemblersprache MACRO-SM

Zeichen	Bedeutung/Funktion
A–Z	Alphazeichen (Großbuchstaben des lateinischen Alphabets)
⊠ **($)**	Zusätzliches Alphazeichen (reserviert für Systemsymbole); Kennzeichen für lokale Symbole
.	Zusätzliches Alphazeichen in Verbindung mit anderen Alphazeichen (reserviert für Systemsymbole); Speicherplatzzuordnungszähler (SZZ) des Assemblers; Dezimalpunkt nach Zahlen
0–9	Numerische Zeichen
:	Symboldefinition, Begrenzer des Namensfeldes
=	Wertzuweisung für ein Symbol oder SZZ
%	Registerkennzeichen
#	Direktwertkennzeichen
@	Kennzeichen der Speicherindirektadressierung
()	Registerindirektadressierung
,	Trennzeichen für Operanden
;	Kommentarkennzeichen
<>	Klammerung für Argumente und Ausdrücke
+	Additionszeichen und Autoinkrementkennzeichen
–	Subtraktionszeichen und Autodekrementkennzeichen
*****	Multiplikationszeichen
/	Divisionszeichen
&	Logisches UND
!	Logisches (inklusives) ODER
"	Kennzeichen für die Definition von zwei ASCII-Zeichen
'	Kennzeichen für die Definition von einem ASCII-Zeichen; Begrenzer für formale Argumente in Makrodefinitionen
^	Universeller einstelliger Operator; Kennzeichen der Pfeilkonstruktion
****	Operator für numerische Symbole im Makroaufruf
?	Kennzeichen für die automatische Generierung eines lokalen Symbols für ein formales Argument, falls keine Substitution durch einen aktuellen Parameter erfolgt
SP	Begrenzer eines einzelnen Symbols oder Feldes
HT	Begrenzer eines einzelnen Symbols oder Feldes
FF	Seitenwechsel

Ein Symbol realisiert eine globale Bezugnahme, wenn es im Programm nicht definiert ist (bei **.ENABL GBL**) bzw. durch eine **.GLOBL**-Anweisung als *global* spezifiziert wird (bei **.DSABL GBL**). In beiden Fällen bekommt das Symbol das Attribut ***global***.

Beispiel:



Nutzersymbole werden außerdem durch ein *Verschieblichkeitsattribut* charakterisiert.
Nichtverschiebliche Symbole haben das Attribut *„absolut"*, verschiebliche das Attribut *„relativ"*. In nichtverschieblichen *Programmabschnitten* (PA) dürfen nur absolute Symbole verwendet werden. In verschieblichen PA dürfen relative und absolute Symbole verwendet werden.
Wird ein Symbol im Namensfeld einer Anweisung definiert (:), dann werden ihm immer der Wert und das Verschieblichkeitsattribut des SZZ zugewiesen. Der SZZ wird in nichtverschieblichen PA immer in absoluten Adressen und in verschieblichen PA immer in relativen Adressen geführt (entspricht dem Verschieblichkeitsattribut des PA).
Ein Symbol, das durch eine Direktanweisung definiert wird, erhält immer das Verschieblichkeitsattribut des Ausdrucks rechts des Gleichheitszeichens (=).
Registersymbole sind die standardisierten Bezeichnungen für die allgemeinen Prozessorregister. Die Registersymbole sind im Assembler permanent definiert.

1.4.2. Lokale Symbole

Lokale Symbole werden aus einer natürlichen Dezimalzahl und dem Sonderzeichen ⊠ gebildet. Die Dezimalzahl darf den Wert von **1** bis **65535** annehmen. Z. B. **1⊠**, **2⊠**, ... **65535⊠**.
Die Symbole **64⊠** bis **127⊠** sind für die Makrogenerierung reserviert.
Der *Gültigkeitsbereich* lokaler Symbole erstreckt sich standardgemäß (**.DSABL LSB**) auf den Bereich zwischen zwei Anweisungen mit einem Symbol im Namensfeld. Für diesen Bereich wird ein lokaler Symbolblock (LSB) aufgebaut, wenn wenigstens ein lokales Symbol definiert wird. Außerdem wird mit jeder **.PSECT**-Anweisung ein neuer LSB eröffnet.
Zur Erhöhung der Effektivität des Assemblers kann der Programmierer den Gültigkeitsbereich der lokalen Symbole steuern und die Kapazität der LSB besser ausnutzen. Ein LSB, der mit einer **.ENABL-LSB**-Anweisung eingerichtet wurde, besitzt seine Gültigkeit bis zur nächsten **.ENABL-LSB-**, **.DSABL-LSB-** oder **.PSECT**-Anweisung.

Das Verschieblichkeitsattribut lokaler Symbole entspricht immer dem Verschieblichkeitsattribut des PA.

1.5. Konstanten

In den Ausdrücken im Operandenfeld der Anweisungen können *Konstanten* benutzt werden, die als Zahlen, Zeichenkonstanten oder Zeichenkettenkonstanten definiert sein können. Alle Konstanten werden intern in eine 16-Bit-Zahl umgerechnet.
Numerische Zeichenketten ohne Angabe der Basis des Zahlensystems werden entsprechend der aktuellen Basis des Zahlensystems interpretiert. Die Basis des Zahlensystems wird durch die **.RADIX**-Anweisung festgelegt. Bei ausgelassener **.RADIX**-Anweisung wird die Basis 8 angenommen. Mit dem universellen einstelligen Operator **^x**, besteht die Möglichkeit, die Basis des Zahlensystems temporär für einen Wert zu ändern:

^B – Binärzahlen: **^B101110,^B100**
^O – Oktalzahlen: **^056,^0177777**
^D – Dezimalzahlen: **^D99,^D32767**

Für den Operator **^D** existiert eine Kurzschreibweise mit nachgestelltem Dezimalpunkt, z. B. **99., 32767.**
Als zusätzliche Operatoren sind die Vorzeichen- und Komplementoperatoren zulässig:
^C – Bildung des Einerkomplements:

^C^B100,^C^056
– – Bildung des Zweierkomplements:
–^B100,–99.,^D–99

Der Operator ± als Vorzeichenoperator ist zulässig, aber redundant.
Zeichen- und Zeichenkettenkonstanten werden mittels der Operatoren **'** und **"** von druckbaren Zeichen des GOST- bzw. ASCII-Kodes gebildet. Der Operator **'** bewirkt, daß das nachfolgende ASCII-Zeichen in den 7-Bit-Kode umgewandelt und als 16-Bit-Wert dargestellt wird:
'A ≙ ^0101,'+ ≙ ^053 usw.

Der Operator **"** bewirkt, daß die zwei nachfolgenden ASCII-Zeichen entsprechend der 7-Bit-Kodetabelle in einen 16-Bit-Wert umgewandelt werden, so daß das erste Zeichen im niederwertigen Byte und das zweite im höherwertigen Byte gespeichert ist:

"AB ≙ ^0041101,"+3 ≙ ^0031453

1.6. Terme und Ausdrücke

Unter einem *Term* versteht man die elementaren syntaktischen Einheiten eines Ausdrukkes. Terme sind:

– Symbole (Nutzer- und permanente Symbole), z. B. **FELD, A0, MARK**
– lokale Symbole, z. B. **54⊠, 23⊠**
– der Speicherplatzzuordnungszähler (SZZ).

Eine Bezugnahme auf den aktuellen Wert des SZZ erfolgt durch einen alleinstehenden Punkt (.), z. B. **.+20, A0–.+2**
– Konstanten (Zahlen mit und ohne Angabe der Basis des Zahlensystems und Zeichenkettenkonstanten), z. B. **^B101, 25, 25., 'A**
– Terme, auf die ein einstelliger Operator angewendet wird, z. B. **^C25, +25, –25., ^C"AB, ^C+25.**
– Terme, die durch spitze Klammern oder die Pfeilkonstruktion geklammert sind, z. B.

<+25>, <^C"AB>, ^/^C"AB/, ^#^C<+25.>#

Die Pfeilkonstruktion **^x** ... *x*, wobei *x* ein beliebiges Zeichen ist, wird analog den spitzen Klammern zur Klammerung von Termen, Ausdrücken und Argumenten benutzt. Das zur Klammerung benutzte Zeichen *x* darf innerhalb der Klammern nicht nochmals verwendet werden. Die Pfeilkonstruktion wird insbesondere in der Makrotechnik angewendet, wo die spitzen Klammern unzulässig oder unübersichtlich sind.

– Ausdrücke, die durch spitze Klammern oder die Pfeilkonstruktion geklammert sind, z. B.

<A0–B>, <'Z–'A–1>, ^Z BETA–25 Z

Die Anwendung mehrerer einstelliger Operatoren auf einen Term ist zulässig. Die Abarbeitung der einstelligen Operatoren erfolgt in der Reihenfolge von rechts nach links.

Beispiel:
Der Term **+–^C^0177776**
entspricht dem Term
<+<–<^C<^0177776>>>>.

Unter einem *Ausdruck* versteht man einen Term oder mehrere Terme, die untereinander durch zweistellige Operatoren verknüpft sind. Zweistellige Operatoren sind:

+ Addition
– Subtraktion
* Multiplikation
/ Division
& logisches UND
! logisches (inklusives) ODER.

Alle zweistelligen Operatoren haben die gleiche Priorität. Die Berechnung der Ausdrücke erfolgt so, daß zuerst alle Terme berechnet werden und anschließend die Werte der Terme durch die zweistelligen Operatoren in der Reihenfolge von links nach rechts verknüpft werden.

Beispiel:

```
     P2 - P1  /  16.  *  16.  +  <FELD ! ^C^ 017>
1.   |        |   |       |      |      | |__| |
2.   |        |   |       |      |      |____| |
3.   |        |   |       |      |___________| |
4.   |________|   |       |                    |
5.   |____________|       |                    |
6.   |____________________|                    |
7.   |_________________________________________|
```

Bei der Ausdrucksberechnung wird für nichtdefinierte Symbole oder fehlerhafte Terme der Wert 0 angenommen. Steht zwischen zwei Termen kein zweistelliger Operator, so werden die Terme links und rechts des Trennzeichens als getrennte Ausdrücke gewertet.

Beispiel:
A+2 MASK1&MASK2

Da zwischen den Termen **2** und **MASK1** kein zweistelliger Operator steht, enthält das Beispiel zwei Ausdrücke, **A+2** und **MASK1&MASK2.**

Die Ergebnisse und Zwischenergebnisse der Ausdrucksberechnung werden intern als 16-Bit-Werte dargestellt. Bei Überschreitung der Stellenzahl werden die Werte linksseitig beschnitten (T-Fehler). Steht ein Ausdruck in einer **.BYTE**-Anweisung, so wird das Ergebnis nach der Ausdrucksberechnung auf 8 Bit verkürzt.
Bei der Ausdrucksberechnung wird das *Verschieblichkeitsattribut* des Ergebnisses ermittelt. Das Ergebnis eines Ausdruckes darf in einem verschieblichen PA absolut oder relativ sein. Ein Ausdruck ist absolut, wenn die Anzahl der relativen Terme mit positiven Vorzeichen gleich der Anzahl der relativen Terme mit negativem Vorzeichen ist.
Ein Ausdruck ist relativ, wenn die Anzahl der Terme mit positivem Vorzeichen um 1 größer ist als die Anzahl der Terme mit negativem Vorzeichen.
Die Multiplikation und Division relativer Terme sowie logische Operationen mit relativen Termen sind unzulässig.

Beispiel:
Es sei angenommen,
daß die Symbole **A, B, C** und **D** relativ und **I, J, K** und **L** absolut sind.
A–B absolut
A–B–C falsch (negativ verschieblich!)
A–B+C relativ
C–D+A+B falsch (doppelt verschieblich!)
I+2–A+B absolut
4*C–D falsch (Multiplikation!)
4*<C–D> absolut
C–D/16*16+<K!L> absolut
A+<I&J&<K!L>> relativ

Sind in einem Ausdruck globale Bezugnahmen oder Bezugnahmen auf Symbole anderer PA vorhanden, so wird die Ausdrucksberechnung teilweise oder vollständig in den Taskbuilder verlagert, wo die Ausdrucksberechnung zur Taskbuilderzeit nach den gleichen Regeln erfolgt. Das Verschieblichkeitsattribut einer globalen Bezugnahme entspricht dem globaldefinierten Symbol im anderen Quellprogramm.

wird fortgesetzt

Literatur

/1/ Merkel, G.: Neue Gerätesysteme des ESER und SKR. rechentechnik/datenverarbeitung, Berlin 16, (1979) Beiheft, S. 2–8
/2/ Programmnoje obespetschenije SM–3. Diskovaja operazionnaja sistema obschtschewo nasnatschenija. Programma Makroassembler. 4.072.088 IE2
/3/ Horn, Th.; Kofer, C.: Handbuch der SKR-Programmierung. Schriftenreihe „Informationsverarbeitung im Hoch- und Fachschulwesen", Heft 1. ZLO Zwickau (1979)
/4/ Horn, Th.; Utke, M.: Programmierung in der Assemblersprache MACRO-SM. Heft 3 der Schriftenreihe „Programmierung von Rechenanlagen des SKR".
Ingenieurhochschule Dresden (1984)



Börse

Anschluß des SD 1154 sowie des LBS 1215 an den KC 85/2

Mit einem relativ geringen Hardwareaufwand wurde für den SD 1154 und den Stanzer 1215 eine TTL-gerechte SIF-1000-Schnittstelle erstellt. Durch den Einsatz eines Eigenbaumoduls für den KC 85/2, verbunden mit spezieller Software, wurde rechnerseitig die SIF-1000-Schnittstelle bewerkstelligt.
Durch zusätzliche Bestückung des SD 1154 mit einem U401 (BM008-kyrillische Zeichen) und eine Erweiterung des KC-85/2-Zeichensatzes um kyrillische Zeichen ist die Bildschirmanzeige und der Ausdruck dieser Zeichen möglich.
Aus- und Weiterbildungszentrum Informatik der Ingenieurschule für Transportbetriebstechnik Gotha, Karl-Marx-Straße 5–7, Gotha, 5800.

Groß

Programmpaket DAZU

DAZU (Dateizugriff) realisiert auf dem KBR A 6402:
- Strukturänderungen der Datei
- Strukturänderungen im Satz
- Berechnungen im Satz bzw. über die ganze Datei
- Dateiformatänderungen
- Dateiprüfungen.

DAZU bietet u. a.:
- einen Eingabebereich (max. 1000 Byte bzw. Zeichen)
- einen Ausgabebereich (max. 1000 Byte)
- einen Zusatzbereich (mit 2000 Byte für die Nutzung als Zwischenspeicher von Rechenergebnissen, Zeichenketten usw.)
- Konvertierungszeiten von Text in Zahlenformat von weniger als 0,045 s.

Gefordert wird lediglich die Vorgabe der Parameter als Textdatei.

Vorteile sind:
- einfache Erstellung
- jederzeit lesbar für den Anwender
- leichte Änderungsmöglichkeit.

VEB Fahrzeugachsen Gotha, Abt. LOO, Südstraße 19, Gotha, 5800; Tel. 5481 App. 301.

Zerbst

Drucker-Interface

Um Drucker des Typs Atari 1020 an verschiedenen Rechnern nutzen zu können, wurde ein Druckercontroller auf Basis U 8820 D entwickelt. Der Controller bedient die DATAOUT-, DATAIN- und COMMAND-Leitungen und wird selbst über ein paralleles oder serielles Interface (PIO- oder V.24-Schnittstelle) angesprochen. Der Controller liest blockweise die Druckzeichen aus dem angeschlossenen Rechner ein und formt daraus entsprechende Daten- bzw. Commandoframes, die er dem Atari 1020 übergibt. Alle Funktionen des Druckers sind vom Bedienrechner (z. B. PC 1715, KC) aus ansprechbar. Die Softwareentwicklung wurde auf einem SCPX-kompatiblen Hard- und Softwareentwicklungsplatz durchgeführt; das Programm liegt als Quell- und Objektdatei auf Diskette bzw. auf EPROM vor.
Hardwareaufwand: U 8820, U 2716, Quarz 7,3728 MHz, IS 75150, IS 75154.
Institut für Züchtungsforschung, BfN, Ethel-und-Julius-Rosenberg-Straße 22/23, Quedlinburg, 4300; Tel. 470.

Kramer

Einzugsvorrichtung für Einzelblätter

An den Druckern, die zu Bürocomputern (z. B. A 5110) mitgeliefert werden, gibt es keine Vorrichtung, die Einzelblattverarbeitung zuläßt. Die angebotene Neuerung enthält die Beschreibung für den Bau einer solchen Einzugsvorrichtung. Das entwickelte Gerät wird an der Druckerabdeckung befestigt, und es besteht die Möglichkeit, sie über die gesamte Walzenbreite zu verstellen.
Martin-Luther-Universität Halle-Wittenberg, Bereich Medizin, Leninallee 5, Halle, 4020.

Kassettenmagnetbandanschluß für SM 4–20

Die Lösung realisiert den hard- und softwaremäßigen Anschluß des Kassettenmagnetbandgerätes robotron K 5261.03 an das Rechnersystem SM 4–20 unter dem Betriebssystem OSRW 2.8. Mit der Kopplung kann im wesentlichen der Anschluß von Bildschirmterminals an der SM 4–20 zur Datenerfassung und der damit verbundene Projektierungsaufwand eingespart werden.
Weitere Vorteile:
- MBK als zusätzlicher Datenträger
- Beibehaltung bisheriger Datenerfassungstechnologie gegeben
- Möglichkeit des Datenaustausches zwischen SM 4–20 und Bürocomputer mit Kassettentechnik.

VEB Werk für Fernsehelektronik Berlin, Werk Mikrooptoelektronik, Bereich Ökonomie und Datenverarbeitung, Abt. HOE 3, Borkumstr. 2, Berlin, 1100; Tel. 4703221 (Koll. Grzeskowiak) oder Tel. 4703310 (Koll. Torge)

Erzeugnisübersichten

Die folgenden Programme für BC A 5130 bzw. PC 1715 dienen der Objektivierung des Entscheidungsprozesses:
- Kombiniertes Programm für die Eingabe, Fortschreibung und Korrektur der Erzeugnisdaten in Betriebsdateien
- Druckprogramme für
  - Erzeugnisübersichten nach Betrieben
  - Gesamtübersicht Kombinat
  - Übersicht nach S- und M-Bilanzen
  - Übersicht zum Erzeugnisumschlag und zu den daraus resultierenden ökonomischen Effekten.

VEB Elektromaschinenbau Dresden, Hennigsdorfer Str. 25, Dresden, 8017

Wir suchen ...

... zur Abarbeitung auf BC/PC ein Materialprojekt für etwa 50000 Positionen. Hinweise bitte an
VE Dienstleistungsbetrieb Berlin, Abt. ODV, Adalbertstraße 49, Berlin, 1040; Tel. 27392404 (Dr. Röhle).

Weidner

Ergänzung zum Beitrag „Mikroprozessorkompatibler D/A-Wandler C 560 D" in MP 5/1987

Wie in dem Artikel auf S. 140 angekündigt, veröffentlichen wir an dieser Stelle die Tafeln 3 und 4.

Tafel 3 Betriebsbedingungen

Kenngröße	Kurzzeichen	Kleinstwert	Größtwert
Betriebsspannung für 2,5 V Endwert	U_{CC}	4,5 V	16,5 V
Betriebsspannung für 10 V Endwert	U_{CC}	11,4 V	16,5 V
Ausgangsstrom	I_o	0 mA	5 mA
L-Eingangsspannung Anschluß 1–10	I_{IL}	0 V	0,8 V
H-Eingangsspannung Anschluß 1–10	I_{IH}	2,0 V	5,5 V
Betriebstemperatur	ϑ_a	0 °C	70 °C

Tafel 4 Elektrische Kennwerte für ϑ_a = 25 °C −5 K

Kenngröße	Kurzzeichen	Bedingungen	Kleinstwert	Größtwert
Endwertausgangsspannung 2,5 V-Bereich	U_o	U_{CC} = +5 V, I_o = 30 µA/5 mA	2,25 V	2,5 V
Endwertausgangsspannung 10 V-Bereich	U_o	U_{CC} = +15 V, I_o = 30 µA/5 mA	9 V	10 V
Offsetfehler (2,5 V-Bereich)	F_o	U_{CC} = +5 V, I_o = 30 µA	−1 LSB	+1 LSB
Linearitätsfehler (2,5 V-Bereich	F_L	U_{CC} = +5 V		0,5 LSB
diff. Linearitätsfehler	$\|F_D\|$	U_{CC} = 5 V		1 LSB
Nebenkennwerte				
Stromaufnahme	I_{CC}	U_{CC} = 16,5 V		25 mA
Betriebsspannungsunterdrückung	$\|SVR\|$	U_{CC} = 4,5 V ... 7 V		0,5 LSB/V

(Fortsetzung von Seite 206)

Sync-zeichen	Zylinder-nummer	Kopf-nummer	Sektor-nummer	Datenblock (512 Byte)	CRC-Zeichen (2 Byte)

Bild 3 Aufbau eines physischen Sektors

Beim Lesen gelangt das Lesesignal zuerst zum Decoder, wo das Takt-Daten-Gemisch des FM-codierten Signals in Daten und Takt getrennt wird. Die Taktrückgewinnung wird rein digital mit einem retriggerbaren Univibrator vollzogen, so daß sich ein Abgleich erübrigt. Die Daten werden anschließend in den seriellen Lesepuffer geschrieben, dann mittels des SIO-Bausteins ausgelesen und in 8-Bit-parallele Daten gewandelt. Der SIO vollzieht dann auch die CRC-Prüfung.
Die Folgesteuerung von SIO und seriellem Puffer erfolgt softwaremäßig über Ausgaben des Prozessors (Bild 2).
Die Baugruppen *Interface-Host* und *Interface-Controller* stellen die Verbindung zwischen dem Wechselplattencontroller und einem Zielrechner dar (siehe Bild 1). Für diese Verbindung wurde eine parallele Schnittstelle gewählt, da sich diese realtiv einfach bei den unterschiedlichsten Mikrorechnersystemen realisieren lassen.
Controller für Winchesterlaufwerke weisen im allgemeinen ebenfalls eine parallele Schnittstelle auf, meist entsprechend dem SCSI- oder dem SASI-Standard /2/, /3/, weshalb sich die beschriebene parallele Schnittstelle an diesen Standards orientiert. Damit wird ein späterer Austausch des Wechselplattencontrollers gegen einen Winchestercontroller vereinfacht.
Das parallele Interface stellt ein asynchrones 8-Bit bidirektionales Businterface dar, bei dem 8 zusätzliche Signale zur Bussteuerung bzw. zur Quittierung dienen. Kernstück der Interfacebaugruppe ist ein PIO-Baustein U 855.
Die Interfacebaugruppe kann durch die Verwendung des PIO sowohl interruptorientiert als auch im Polling eingesetzt werden. Wird als Zielrechnersystem ebenfalls ein K-1520- oder ein K-1520-kompatibles System (U-8000-Erweiterungsmodul des Bürocomputers A 5120.16) verwendet, so sind die Baugruppen Interface-Host und Interface-Controller nahezu identisch.
Anderenfalls muß eine ähnliche Baugruppe entsprechend dem Systembus des Zielrechners unter Verwendung des PIO oder anderer geeigneter paralleler E/A-Bausteine, z. B. 8255, für 8086-Systeme verwendet werden.
An den Controller sind zwei Wechselplattenspeicher EC 5052 anschließbar. Damit steht ein Massenspeicher von 12 MByte zur Verfügung, womit alle Back-up-Probleme gelöst sind.

3. Softwarekomponenten

Die Software für den Wechselplattencontroller wurde mit dem Betriebssystem UDOS entwickelt. Die einzelnen Programmteile sind so gehalten, daß sie problemlos die Wechselplatte direkt oder als abgesetzte Einheit steuern können. Unter dem Betriebssystem SCP ist die Wechselplatte direkt angeschlossen. Mit dem abgesetzten Controller wird die Wechselplatte unter den Betriebssystemen MUTOS (Interruptbetrieb) und Single-User-Betriebssystem für 8086-Mikrorechner (Pollingbetrieb) verwendet. Die Parameter- und Datenübergabe erfolgt mit einem Protokoll in Anlehnung an /4/. Als Parametervektor werden 6 Byte übergeben:

Klaus Graumann (33) studierte von 1974–1978 an der Wilhelm-Pieck-Universität Rostock technische Kybernetik und Automatisierungstechnik.
Von 1978 bis 1982 wissenschaftlicher Assistent, seit 1982 ist er F/E-Ing. im Wissenschaftsbereich Mikrorechentechnik/Schaltungstechnik an der Sektion Technische Elektronik der WPU.

Klaus Kopplow (43) studierte von 1961 bis 1966 Physik in Rostock und arbeitete bis 1981 an der Datenverarbeitung.
Seitdem ist er als Ingenieur für Forschung an der Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, im Wissenschaftsbereich Mikrorechentechnik/Schaltungstechnik tätig.

- Kommando (1 Byte)
- Laufwerk (1 Byte: Bit 5 ... 7)
- Blocknummer (2 Byte)
- Byteanzahl (2 Byte).

Anstelle der Byteanzahl wird in einigen Kommandos die Blockzahl angegeben. Folgende Kommandos sind möglich:

- Initialisieren des Controllers
- Einstellen einer Spur
- Lesen bzw. Schreiben einer Anzahl von Datenbyte
- Formatieren einer Blockanzahl (ab Spuranfang)
- Kopieren einer Blockanzahl (2 Wechselplatten).

Zur Prüfung der Wechselplatte und der Hardware wurden weitere Kommandos vorbereitet.
Wird bei der Ausführung eines Kommandos ein Fehler festgestellt, so wird er in einem Byte verschlüsselt abgelegt und bei Operationsende zusammen mit dem Laufwerk und der logischen Blocknummer nach der Statusübergabe automatisch übergeben. Die übergebenen 4 Byte können außerdem in einem Fehlerringpuffer im Speicher abgelegt werden. Als Fehler werden die Meldungen der Wechselplatte, Ablauffehler und formale Fehler erkannt.
Um einen Interruptfehler des Interfaces zu behandeln, kann eine Zeitüberwachung programmiert werden. Sie ist zur Zeit nicht vorgesehen. Bei einer Fehlererkennung werden vom Treiber mehrere Leseversuche bzw. Schreibversuche ausgeführt, bei Positionierfehlern erfolgt keine Wiederholung.
Da die Hardsektorierung der Wechselplatte für die Sektorerkennung ausgenutzt wird, entfällt ein abgesetztes Identitätsfeld. Es wurde für Kontrollzwecke zusätzlich zu den 512 Datenbyte die Sektoradresse (Zylinder, Kopf, Sektor) mit abgespeichert.
Bei dieser Sektorgröße können 6 Sektoren pro Spur abgelegt werden. Da auf einen Sektorsignalzähler verzichtet wurde, beginnt der Suchlauf für einen Sektor ab Indexmarke. Als Synchronbyte wurde 07H gewählt (Bild 3).
Für das Interface zum Hostrechner werden zwei PIO-Tore verwendet. Die Daten werden im Bitmodus übergeben. Die Übergaberichtung wird durch Initialisierung vor jedem Datenblock eingestellt. Die Steuersignale werden über das zweite PIO-Tor ausgegeben bzw. eingelesen (Bitmodus). Der Controller selbst arbeitet nur im Polling, um die Steuerung der Datenübergabe nicht abzugeben und die Bedeutung der Interfacesignale /5/ weitgehendst beizubehalten:

BUSY	Controller arbeitet
I/O	Datenrichtungssignal
C/D	Signal für den Dateninhalt
MSG	Information für das letzte Datenbyte
RDY	Controller bereit zum Datenaustausch
SEL	Anforderung vom Host für eine Operation
RST	Anforderung vom Host für Controllerreset
ACK	Bestätigung vom Host für Datenaustausch.

Der Hostrechner kann die Signale für einen Interrupt- oder Pollingbetrieb nutzen. Die Datenübergabe unter einem Single-User-Betriebssystem für 8086-Mikrorechner wurde im Pollingbetrieb mit dem Baustein 8255 realisiert. Um den Controller an die logische Sektorgröße des Betriebssystems anzupassen, wird bei der Initialisierung die Blockbezugsgröße angegeben (Vielfaches von 2 exp.n * 128). Damit kann der Controller die Daten der logischen Blocknummer zur Verfügung stellen.
Bei der Arbeit mit der Wechselplatte unter verschiedenen Betriebssystemen wurde ein Geschwindigkeitszuwachs um einen Faktor 2 bis 3 im Vergleich zu Disketten festgestellt.

4. Zusammenfassung

In der Sektion Technische Elektronik wurde der Controller an einem 8086-Applikationsrechner mit einem Single-User-Betriebssystem und am Steuerrechner eines vernetzten Laborsystems von 8 Bürocomputern unter MPM eingesetzt; ebenso am BC A 5120.16 unter dem Betriebssystem MUTOS 8000 – hier mit Unterstützung von Kollegen des Zentralinstituts für Kybernetik und Informationsprozesse Berlin bei der Implementierung der Treibersoftware für den Controller. Die Leistungsfähigkeit dieser Systeme konnte dadurch zum Teil beachtlich erhöht werden. Dennoch stellt der Einsatz von Wechselplattenspeichern EC 5052 bezüglich Größe der Geräte, Geräuschbelastung und Wartung als auch der Leistungsaufnahme einen Kompromiß dar. Die Lösung kann und soll Winchesterlaufwerke mit entsprechenden Controllern auf die Dauer nicht ersetzen.

Fortsetzung auf S. 213

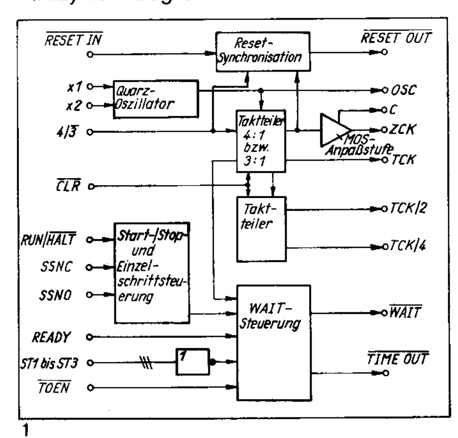
⊠ KONTAKT ☏

Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik, Wissenschaftsbereich Mikrorechentechnik/Schaltungstechnik,
Albert-Einstein-Straße 2, Rostock 6, 2500;
Tel. 45215 oder 45366

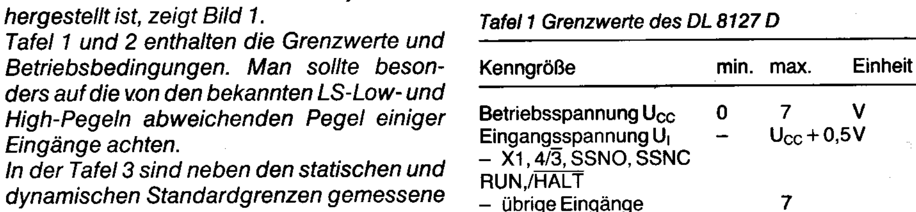
Integrierter Systemtaktgenerator DL 8127 D

Dr. Eckhard Fehse
VEB Halbleiterwerk Frankfurt (Oder)

Die integrierte Schaltung DL 8127 D enthält einen Systemtaktgenerator, der speziell für die Taktversorgung von U-8000- oder 8086-Mikrorechnersystemen ausgelegt ist.
Neben dem Taktoszillator enthält die IS Frequenzteiler, Takttreiber, Kontrollfunktionen wie RUN/HALT, Einzelschritt, RESET und READY sowie einen TIMEOUT-Zähler, der Warteaufforderungen der Peripherie auf 16 Taktzyklen begrenzt. Das Blockschaltbild der IS, die sich in einem 24poligen schmalen (7,5 mm Reihenabstand) Plastgehäuse befindet und in Low-Power-Schottky-Technik hergestellt ist, zeigt Bild 1.
Tafel 1 und 2 enthalten die Grenzwerte und Betriebsbedingungen. Man sollte besonders auf die von den bekannten LS-Low- und High-Pegeln abweichenden Pegel einiger Eingänge achten.
In der Tafel 3 sind neben den statischen und dynamischen Standardgrenzen gemessene typische Werte angegeben.

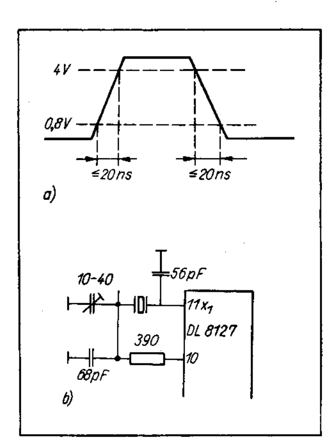


1

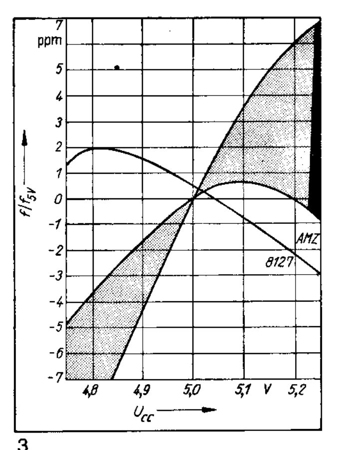


Oszillator

Der Oszillator kann durch eine Quarzschaltung erregt oder mit TTL-Signal am Pin 11 (X_1) fremd gespeist werden.
Die Beschaltung des Oszillators ist frequenzabhängig. Im Frequenzbereich 4 ... 20 MHz kommen Grundwellenquarze zur Anwendung. Die Beschaltung zeigt Bild 2b.
Der mit ∗ gekennzeichnete Kondensator kann mit steigender Frequenz kleiner werden und oberhalb 10000 kHz wegfallen.
Grundwellenquarze in Serienresonanz mit 30 pF Last ausgemessen, lassen sich in der in Bild 2b vorgegebenen Oszillatorschaltung auch auf die angegebenen Quarzfrequenzen abgleichen. Die tendenzielle Abhängigkeit der Frequenz über der Betriebsspannung ist in Bild 3 dargestellt. Oberhalb der empfohlenen Betriebsspannung (5 V) ist die Abhängigkeit geringer.
Der typische TK der internen Oszillatorschaltung liegt im Bereich von 25 °C bis 70 °C bei etwa −4 ppm/grd.
Für höhere Oszillatorfrequenzen müssen Oberwellenquarze eingesetzt werden. Die



2



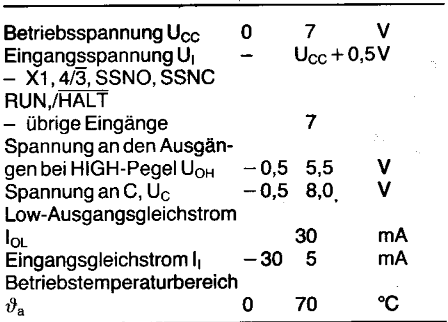
3

Bild 1 Blockschaltbild des DL 8127 D

Bild 2 a) Spezifikation des MOS-Taktes bei 200 pF Last
b) Quarzbeschaltung für den Frequenzbereich 4000 bis 20000 kHz

Bild 3 Betriebsspannungsabhängigkeit der Oszillatorfrequenz

Tafel 1 Grenzwerte des DL 8127 D



Kenngröße	min.	max.	Einheit
Betriebsspannung U_{CC}	0	7	V
Eingangsspannung U_I – X1, $4/\overline{3}$, SSNO, SSNC RUN,/$\overline{HALT}$	–	U_{CC} + 0,5	V
– übrige Eingänge		7	
Spannung an den Ausgängen bei HIGH-Pegel U_{OH}	−0,5	5,5	V
Spannung an C, U_C	−0,5	8,0	V
Low-Ausgangsgleichstrom I_{OL}		30	mA
Eingangsgleichstrom I_I	−30	5	mA
Betriebstemperaturbereich ϑ_a	0	70	°C

Tafel 2 Betriebsbedingungen des DL 8127 D

Kenngröße	min.	max.	Einheit
U_{CC}	4,75	5,25	V
U_{IH} ST1, ST2, ST3, X1 $\overline{CLR}$, $\overline{TOEN}$, READY	2		V
RUN/$\overline{HAL}$, SSNO, SSNC, $4/\overline{3}$	2,4		V
$\overline{RESET IN}$	2,8		V
U_{IL} ST1, ST2, ST3, X1 $\overline{CLR}$, $\overline{TOEN}$, READY RUN/$\overline{HALT}$, SSNO, SSNC, $4/\overline{3}$		0,8	V
$\overline{RESET IN}$		0,4	V
$-I_{OH}$ ZCK		0,2	mA
TTL-Ausgänge		2,6	mA
I_{OL} ZCK		2	mA
TTL-Ausgänge		16	mA
t_s $\overline{CLR}$ --- OSC		25	ns
t_H $\overline{CLR}$ --- OSC		−6	ns
t_s READY --- ZCK $4/\overline{3}$ = High	T/4 + 10[1]		ns
$4/\overline{3}$ = Low	20		ns
t_H READY --- ZCK $4/\overline{3}$ = High	−T/4[1]		ns
$4/\overline{3}$ = Low	−5		ns
t_s ST1, ST2, ST3 --- ZCK $4/\overline{3}$ = High	T/4 + 12[1]		ns
$4/\overline{3}$ = Low	25		ns
t_H ST1, ST2, ST3 --- ZCK $4/\overline{3}$ = High	− (T/4 − 3)[1]		ns
$4/\overline{3}$ = Low	−12		ns
t_s $\overline{TOEN}$ --- ZCK $4/\overline{3}$ = High	35		ns
$4/\overline{3}$ = Low	30		ns
t_H $\overline{TOEN}$ --- ZCK $4/\overline{3}$ = Hig	−15		ns
$4/\overline{3}$ = Low	−10		ns

Anmerkung:
t_s = Einstellzeit
t_H = Haltezeit
[1]T = ZCK-Taktperiodendauer

Fortsetzung von S. 212

Literatur

/1/ Handbuch ESER-Geräte. Abschnitt A6 WPS EC 5052, Seite 1–23. VEB Robotron-Vertrieb Leipzig, Schulungszentrum
/2/ Zeltwanger, H.: Ein Peripheriegerätebus setzt sich durch. Elektronik 1986, Heft 18, Seite 34–36
/3/ Killmon, P.: For computer systems and peripherals, smarter is better. Computer Design 1986, Januar 15, Seite 57–70
/4/ Anschluß-Controller (AC). Betriebsdokumentation. VEB Kombinat Robotron 1983
/5/ XEBEC S 1410 5,25 Inch Winchester Disk Controller. Owner's Manual. Xebec systems, Incorporated 1982

Interrupt

Fast wie ein Interrupt für die redaktionelle Arbeit unserer Zeitschrift MP wirkt z. Z. die unbesetzte Stelle einer Redaktionssachbearbeiterin bzw. eines -sachbearbeiters.
Falls Sie Interesse an dieser Tätigkeit haben, gerne mit moderner Technik arbeiten und im Raum Berlin wohnen, rufen Sie uns unter Tel. 2870203 oder 2870371 an.
Gleichzeitig möchten wir all jene Leser, die uns geschrieben haben, dafür um Verständnis bitten, daß Sie etwas länger als gewohnt auf unsere Antwort warten müssen.

Ihre Redaktion MP

Tafel 3 Spezifische Kennwerte

Kenngröße	Bedingungen	statische Kennwerte min.	typ.	max.	Einheit
I_{IH}	U_{CC} = 5,25 V				
4/$\overline{3}$, SSNC, SSNO	U_{IH} = 2,7 V				
$\overline{\text{RUN/HALT}}$		−300	−120		μA [1]
$\overline{\text{RESET IN}}$		−200	−50		μA [1]
ST1, ST2, ST3					
$\overline{\text{CLR}}$, $\overline{\text{TOEN}}$, READY			1	50	μA
X1			220	600	μA
I_{IH}	U_{CC} = 5,25 V				
ST1, ST2, ST3	U_{IH} = 5,5 V				
$\overline{\text{CLR}}$, $\overline{\text{TOEN}}$, READY			1	1000	μA
$-I_{IL}$	U_{CC} = 5,25 V				
SSNO	U_{IL} = 0,4 V		0,900	1,6	mA
SSNC, 4/$\overline{3}$, RUN/$\overline{\text{HALT}}$					
READY			0,390	1,2	mA
$\overline{\text{CLR}}$, $\overline{\text{TOEN}}$, X1			0,310	0,72	mA
ST1, ST2, ST3					
$\overline{\text{RESET IN}}$				0,36	mA
U_{OH}	U_{CC} = 4,75 V				
ZCK	$-I_{OH}$ = 0,1 mA	4,35		4,6	V
	$-I_{OH}$ = 0,2 mA	4,00	4,5		V
TTL-Ausgänge	$-I_{OH}$ = 2,6 mA	2,4	3		V
U_{OL}					
ZCK	I_{OL} = 2,0 mA		0,29	0,4	V
TTL-Ausgänge	I_{OL} = 16 mA		0,32	0,5	V
$-I_{OS}$	U_{CC} = 5,25 V [2]				
ZCK		50	100	240	mA
TTL-Ausgänge		40	60	130	mA

Kenngröße	Bedingungen	dynamische Kennwerte min.	typ.	max.	Einheit
I_{CC}	U_{CC} = 5,25 V X1 = 2,4 V ZCK = Low TCKs = Low [3]		80	140	mA
t_{TLH} ZCK	C_L = 80 pF ± 5 %		10	12	ns
t_{TLH} ZCK			6	11	ns
t_{TLH} ZCK	C_L = 200 pF ± 5 %		15	20	ns
t_{TLH} ZCK			11	20	ns
t_{PLH} READY --- $\overline{\text{WAIT}}$	U_{CC} = 5 V R_L = 500 Ohm		11	16	ns
t_{PHL} READY --- $\overline{\text{WAIT}}$	C_L = 50 pF		13	19	ns
t_{PLH} ST1, ST2, ST3 --- $\overline{\text{WAIT}}$			15	26	ns
t_{PHL} ST1, ST2, ST3 --- $\overline{\text{WAIT}}$			17	24	ns
t_{PLH} ZCK --- $\overline{\text{RESETOUT}}$	4/$\overline{3}$ = High		10	20	ns
t_{PHL} ZCK --- $\overline{\text{RESETOUT}}$			5	10	ns
t_{PLH} ZCK --- $\overline{\text{RESETOUT}}$	4/$\overline{3}$ = Low		16	20	ns
t_{PHL} ZCK --- $\overline{\text{RESETOUT}}$			8	10	ns
f_{OSZ}			40	24	MHz

Anmerkung:
[1] Negative High-Eingangströme werden durch internen Pull-up-Widerstand verursacht.
[2] Nicht mehr als einen Ausgang gleichzeitig; Dauer des Kurzschlusses 1 s
[3] 1. CLR = Low, SSNO = Low, restliche Eingänge High
2. X1 = 1 Takt (Low/High-Flanken)
3. CLR = High
4. X1 = weitere 15 Takte mit Schlußpegel 2,4 V, SSNO = High
5. Messung von I_{CC}

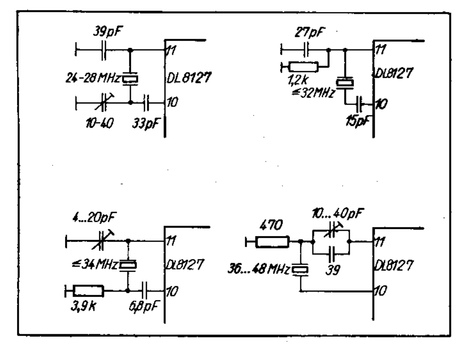


Bild 4 Quarzbeschaltung bei Einsatz von Oberwellenquarzen und DL 8127

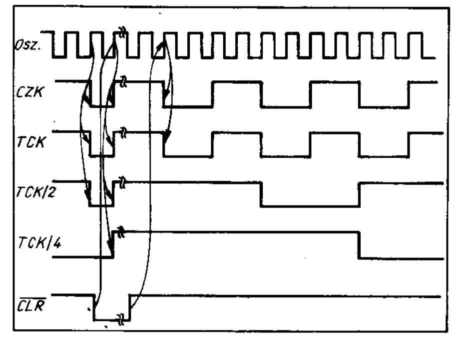


Bild 5 Taktdiagramm U-8000-Modus

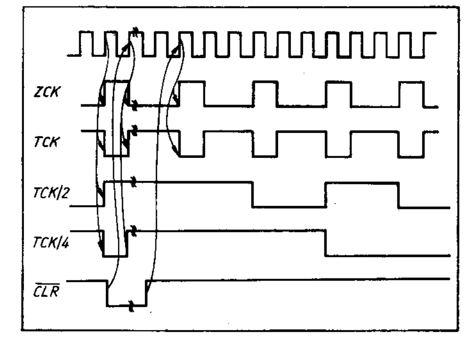


Bild 6 Taktdiagramm 8086-Modus

Schwierigkeit beim Einsatz von Oberwellenquarzen besteht darin, die Oszillatorschaltung so zu wählen, daß der Quarz nur auf der vorgesehenen Oberwelle schwingt.
Die Schaltungen für die Frequenzbereiche 24.. 48 MHz sind in Bild 4 dargestellt. Hierzu muß nachgetragen werden, daß vom VEB HFO nur eine Oszillatorfrequenz von 24 MHz garantiert wird. Die typische maximale Oszillatorfrequenz liegt derzeit bei 40 MHz, einzelne Muster arbeiten bis 48 MHz.

Eckhard Fehse (33) studierte nach der Facharbeiterausbildung als Schlosser von 1972 bis 1976 an der Technischen Hochschule Ilmenau, Sektion Informationstechnik und Theoretische Elektrotechnik.
Von 1976 bis 1980 arbeitete er an der gleichen Sektion als wissenschaftlicher Assistent auf dem Gebiet der Hybridtechnik. Mit einer Arbeit über Rauschen und Nichtlinearität von Dickschichtwiderständen promovierte er 1981 zum Dr.-Ing.
Seit 1980 ist er im HFO tätig und für die Applikation digitaler Schaltkreise zuständig.

Ein extern erzeugter Takt wird mit TTL-Pegel am Anschluß 11 eingespeist. Die Oszillatorfrequenz steht gepuffert am Ausgang Pin 9 zur Verfügung.
In Abhängigkeit vom logischen Pegel am Eingang 3/4 (Pin 13) liegt die Oszillatorfrequenz durch 3 oder 4 geteilt am Ausgang ZCK an, der die Anforderungen für die MOS-Taktversorgung gewährleistet (Bild 2a).
Im U-8000-Modus wird der durch 4 geteilte Oszillatortakt an den Ausgängen CK und TCK mit einem Tastverhältnis 1:1 zur Verfügung gestellt. An den Ausgängen TCK/2 und TCK/4 wird der Systemtakt (ZCK), durch 2 und 4 geteilt, bereitgestellt (Bild 5). Für den 8086 ist ein unsymmetrisches Tastverhältnis erforderlich (Bild 6).
Der Eingang 3/4 wird dazu auf L gelegt. Um die Synchronisation aller Taktausgänge zu ermöglichen, können die Teiler mit $\overline{\text{CLR}}$ taktsynchron rückgesetzt werden.

Spezielle Logikschaltungen und Anwenderschaltung

Die im DL 8127 enthaltenen Logikschaltungen sind an die genannten Mikrorechnersysteme angepaßt, können aber auch anderweitig genutzt werden. Am Anschluß 14 ($\overline{\text{RESETOUT}}$) wird für die CPU ein taktsynchrones (L-H-Flanke) Rücksetzsignal bereitgestellt. Das Pin 15 ($\overline{\text{RESET IN}}$) wird dazu mit einem Kondensator gegen Masse beschaltet. Dieser Eingang hat Schmitt-Trigger-Verhalten. Die minimal zulässige Eingangs-H-Spannung ist 2,8 V. Der typische Schaltpunkt U_{IH} liegt bei 2,5 V mit einer Hysterese von 1 V. Der $\overline{\text{RESETOUT}}$-Impuls muß für U-8000-Systeme mindestens 5 Taktzyklen betragen. Der Kondensatorwert läßt sich in etwa nach der Beziehung (1) ermitteln:

$C \approx 1.1/Fc$ (1)
C in nF
Fc = Systemtakt in MHz
Liegt Pin 13(3/4) auf L, ist RESETOUT H-aktiv.

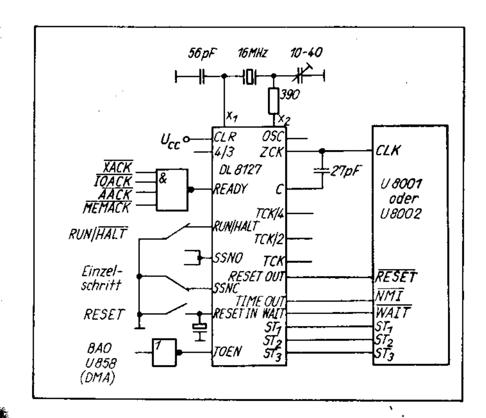


Bild 7 Zusammenschaltung des DL 8127 mit dem U 8001/U 8002

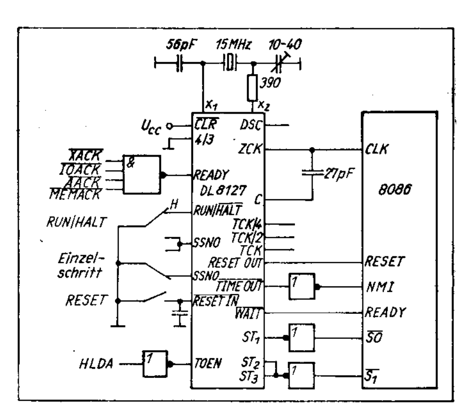


Bild 9 8086-Applikation des DL 8127

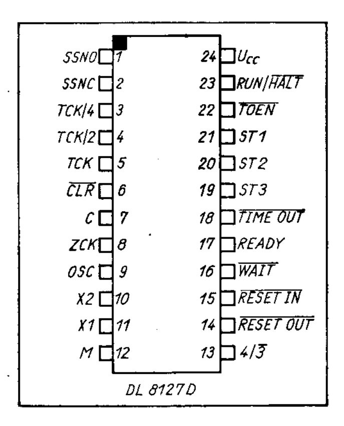


Bild 10 Anschlußbelegung des DL 8127

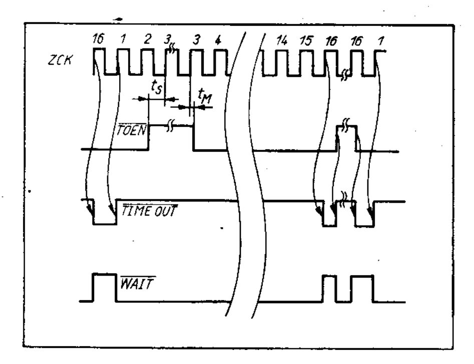


Bild 8 TIMEOUT-Zeitverhalten

Eine weitere Logikeinheit beeinflußt den WAIT-Ausgang, der die CPU veranlaßt, in den Wartezustand zu gehen.
Der WAIT-Ausgang wird auf Low gesetzt, wenn an den Eingängen RUN/HALT oder READY L-Pegel anliegt. Beeinflußt werden kann in diesem Zustand der WAIT-Ausgang durch die Steuereingänge ST1, ST2, ST3. Liegt an allen Steuereingängen L – das bedeutet, daß die CPU interne Operationen oder Speicherauffrischungen durchführt –, wird der WAIT-Ausgang auf H gesetzt, um der CPU diese Operationen zu ermöglichen. Für eine Taktperiode (bei RUN)/(HALT auf Low) geht der WAIT-Ausgang auf H-Potential, wenn am Eingang SSNC ein L-Impuls angelegt wird. Auf diese Weise kann man die CPU quasi im Schrittbetrieb laufen lassen.
SSNC und SSNO sind Eingänge eines Flip-Flops und sollten für Schrittbetrieb mit einem zweipoligen Umschalter beschaltet werden (siehe auch Bild 7). Das WAIT-Signal kann nach 15 CPU-Taktzyklen (trotz L am Eingang READY) aufgehoben werden, wenn zuvor der TIMEOUT-Zähler in Betrieb gesetzt wurde (TOEN auf Low).
Mit der steigenden Flanke des 16. MOS-Taktes wird für eine Taktperiode WAIT auf High und TIMEOUT auf Low gesetzt und damit die WAIT-Forderung eines externen Gerätes begrenzt (Bild 8). Zu beachten ist, daß der Eingang TOEN mit dem Takt synchronisiert werden muß, da sonst der TIMEOUT-Impuls verkürzt wird und doppelt auftreten kann, wenn TOEN während des 16. Taktimpulses auf High geht.
Ersichtlich ist dieser Fall aus Bild 8 (rechte Bildhälfte). Schaltet TOEN während der übrigen Taktperioden auf High, wird der Zählvorgang ausgesetzt. Dies kann z.B. bei DMA-Betrieb der Fall sein. Ebenfalls ausgesetzt wird der Zählvorgang, wenn WAIT auf High geht ($ST_{1,2,3}$ = L). In den Bildern 7 und 9 sind Einsatzbeispiele für die U-8000- und 8086-CPU angegeben.
Die Betriebsspannung des DL 8127 sollte mit 47 nF Scheibenkondensatoren und 47 μF Elkos abgeblockt werden.
Bild 10 zeigt die Anschlußbelegung des Schaltkreises.
Der DL 8127 D kann selbstverständlich auch außerhalb der genannten Rechnersysteme zur Erzeugung eines quarzstabilen Taktes im angegebenen Frequenzbereich verwendet werden. Die Steuerung des 3/4-Eingangs durch TCK/2, TCK/4 oder extern ermöglicht zusätzliche, vor allem ungerade Teilerverhältnisse.

Literatur

/1/ Fachbereichsstandard TGL 43298
/2/ Systemtaktgeneratorschaltkreis DL 8127 D

✉ KONTAKT ☏
VEB Halbleiterwerk Frankfurt (Oder), Abt. EECS, PF 379, Frankfurt, 1200; Tel. 463264.

Echtzeit-Debugger DRTC8000

Wolfgang Rehm
Technische Universität Karl-Marx-Stadt,
Sektion Informationstechnik

Einführung

Zur Lösung industrieller und nachrichtentechnischer Echtzeitsteuerprobleme auf der Basis von Mikrorechnern wurde an der Technischen Hochschule Karl-Marx-Stadt das Programmiersystem PLZ/RTC /1/ geschaffen und in der Folgezeit insbesondere in Verbindung mit der Implementierung auf 16-Bit-Mikrorechnern weiterentwickelt /2/ ... /7/.
Entsprechend dem Grundkonzept werden die Anwenderprogramme in der höheren Echtzeitsprache PLZRTC formuliert, und im Ergebnis eines in mehreren Schritten ausgeführten Übersetzungs- und Bindelaufs erfolgt die Zielkodegenerierung für einen konkreten Prozessor.

Bei der Implementierung für den Prozessor U8000 wurde besonderer Wert auf die Möglichkeit einer effektiven Programmierung auf allen Sprachniveaus gelegt. Die differenzierte Leistungsfähigkeit der unter dem Betriebssystem UDOS laufenden Cross-Software ist dabei ebenso in Betracht gezogen worden wie die Einsatzerfahrung der 8-Bit-Implementation.
Als wesentliches Produkt des Generierungsvorganges entsteht der Echtzeitsteuerkern. Während er vom Gesichtspunkt der Hochsprache im Sinne einer Laufzeitumgebung weniger Beachtung findet, besitzt er für den Assemblerprogrammierer eine zentralere Bedeutung, speziell zum Zeitpunkt des Tests des Programmsystems.

Für den Test eines parallelen Anwenderprogrammpaketes sind übliche Debugger /8/ vollkommen ungeeignet, insbesondere was die Funktionen zur Steuerung des Programmlaufs betrifft. Das leuchtet sofort ein, wenn man bedenkt, daß es nicht nur ein Programm schlechthin gibt, sondern viele (quasi-) parallel ablaufende Prozesse. So kann die Fortsetzung eines Programms beispielsweise nicht einfach durch ein Sprungkommando zu einer definierten Speicherzelle vom Bediener festgelegt und somit die Schedulersteuerung umgangen werden. Eine Unterbrechung darf andererseits nur gezielte Teile eines Prozeßsystems stillegen. Aufgrund der beschränkten Symbolfähigkeit der Cross-Software wurde der Debugger so konzipiert, daß er in Verbindung mit den jeweiligen Programm-(Assembler-)Listen einen effektiven Test des Anwenderprogrammsystems ermöglicht.

Bevor näher auf den Echtzeitdebugger eingegangen wird, soll nachfolgend eine Kurzbeschreibung der RTC-Kernimplementierung RTC8000 /5/ für die 16-Bit-Prozessoren U8001/U8002 gegeben werden.

Tafel 1 Kernruf-Primitiven

Kernruf	Übergabeparameter par 1	par 2	par 3	Rückgabeparameter par 1	par 2	par 3
START	pnr	message1	message2	pnr		
STARTI	pnr	message1	message2			
STOP						
STOPI	pnr					
THIS				pnr-index		
HOLD	pnr					
CONTN	pnr					
SEND	pnr	message1	message2			
WAIT				pnr	message1	message2
CLASS	pnr	class				
PRTY	pnr	priority				
ASSIGN	iveknr					
RESIGN	iveknr					
ENABLE	pnr					
DISABL	pnr					
TIMER	action	zeitwert				
TIMDEL						
RTIME	zeitwert					
STIME	zeitwert					
TIMEOUT				Abbruchkode		
STATUS				Prozeßstatus		
LOCK	Schedulersperre					
UNLOCK	Freigabe					
INTRET	spez. Kerneintritt					

pnr: Prozeßnummer

Tafel 2
Typische Kernreaktionszeiten einiger Kernrufe

Kernruf	Prozeßumschaltung ohne	mit
START	179 + 10∗p	215 + 10∗p
STOP	144	
HOLD	103	142
CONTN	74	130
CLASS	80	178 + 10∗p
PRTY	81	142 + 10∗p
SEND	165	202 + 10∗p
WAIT	82	135
TIMER	118 + 15∗p	
THIS	65 + 10∗p	

p Prozeßanzahl in div. Warteschlangen

RTC8000-Kern

Der RTC8000-Kern ist besonders zur Implementierung von Programmpaketen zur Steuerung mittlerer Prozeßsysteme geeignet.

Er stellt eine qualitative Weiterentwicklung des RTC80-Kerns /1/ dar und wurde äußerlich so gestaltet, daß die im Rahmen-(Sprach-)Konzept /9/ geforderten Funktionen als semantische und syntaktische Untermenge im wesentlichen enthalten sind.

Neu sind unter anderem die Hinzunahme weiterer, leistungsfähiger Kernrufprimitiven zur Zeitverwaltung, Schedulersteuerung und Prozeßfeldverwaltung, die Modifikation der SEND/WAIT-Primitiven im Sinne einer eindeutigen Prozeßkommunikation, die Verfeinerung des Unterbrechungs- (Vorpla-

```
Phil process array[5] (thisis instance mes1,mes2 word)
status:=start
local   x         word
   idx      word
entry
idx:=THIS
if idx=0 then phil0:=thisis
else
CLASS (phil0+instance idx , byte (%78 - 2* idx) )
fi
idx:=%1000 + idx * %800
TIMER (t_contn_cycle, word idx)
!philosophers  life-loop        !
do
   wthinking
   hold_this
   weating
   pickup
   beating
   hold_this
   putdown
od
end Phil

Fork process array[5] (thisis instance mes1,mes2 word)
status:=start
local    fork_status,request,x word
         quelprc,waitprc    instance
         idx      word
entry
fork_status:=0
waitprc:=0

idx:=THIS
if idx=0 then fork0:=thisis fi
do
   quelprc,request,x:=WAIT
   if request=take then
        if fork_status=free then
           fork_status:=notfree
           SEND(quelprc,0,0)
        else
           waitprc:=quelprc
        fi
   else
        if waitprc=0 then
           fork_status:=free
        else
           SEND(waitprc,0,0)
            waitprc:=0
        fi
   fi
od
end Fork
```

Bild 1
Programmauszug „The Dining Philosophers“

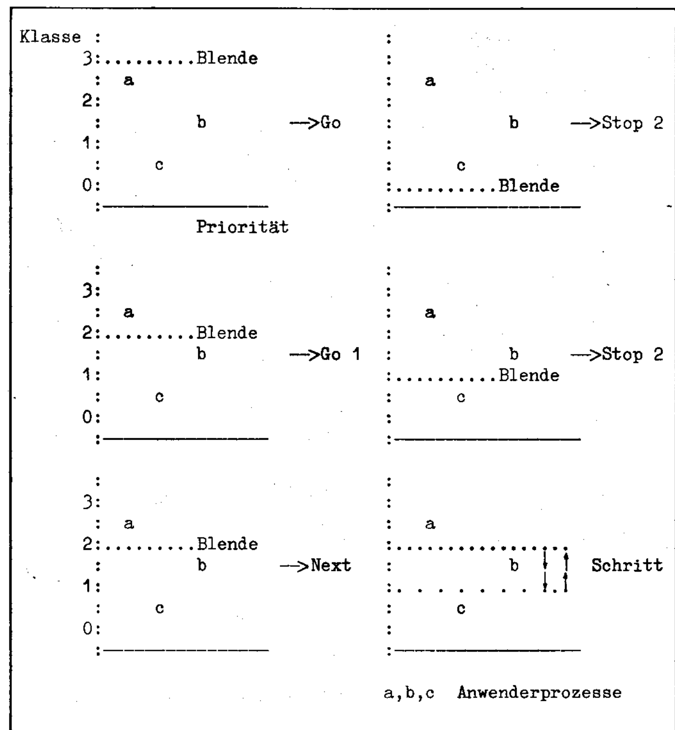


Bild 2
Teststrategie

nungs-) Mechanismus sowie die Unterstützung der Multiprozessor-Kommunikation in verteilten Systemen.
Eine Übersicht über die Kernrufprimitiven und die entsprechenden Kernverwaltungszeiten sind in den Tafeln 1 und 2 gegeben.
Einfachheit und Übersichtlichkeit des RTC-Systems wird durch die Reduktion auf einen einzigen Objekttyp, den *Prozeß* erreicht.
Die zur Kommunikation erforderlichen Puffer (mailboxes) sind implizit mit der Existenz eines jeden Prozesses gegeben und an diesen gebunden. Die Anzahl der Prozeß-Objekte wird (statisch) zum Programmgenerierungszeitpunkt festgelegt.
Triviale Synchronisationsobjekte (Semaphore) werden durch speziell programmierte Prozesse gleichsam effektiv realisiert.
Die Umgebung des Kerns kennt also nur *Prozesse,* deren koordiniertes Multitasking durch diese selbst in Gestalt von Kernrufen gesteuert wird. Zur Verdeutlichung ist in Bild 1 ein Auszug aus einem in der Echtzeit-Hochsprache PLZRTC notierten Programmbeispiel „The Dining Philosophers" gegeben.
Unterbrechungen (Interrupts) werden in Kernrufaktionen überführt. Die eindeutige Verbindung zwischen Interruptvektor und gewünschtem Kernruf wird durch einen Vorplanungsruf erzeugt. Allerdings müssen Unterbrechungen nicht unbedingt in einen Kernruf münden. Vielmehr liegt die Entscheidung darüber in der Interruptroutine selbst. Damit werden auch zeitkritische Unterbrechungsarbeiten systematisch unterstützt.
Ausnahmebehandlungen (Traps) werden wie „normale" Interrupts behandelt, womit die Ausnahmereaktion völlig in der Hand des Programmierers liegt. Der exklusive Zugriff auf kritische Regionen kann auf zweifache Art und Weise realisiert werden. Allgemein wird ein mit der Region logisch verbundener Semaphorprozeß vorgesehen und abgefragt.
Zeitlich kurze kritische Abschnitte sind effektiver durch die Einklammerung mit den Kernoperationen LOCK/UNLOCK zu realisieren. (Im Gegensatz zur adäquaten Realisierung per Kernrufe ENABLE/DISABL können zeitkritische Interruptbursts weiterhin bearbeitet werden.)
Zum Geben und Überwachen von Zeitwerten kann die Systemuhr vermittels definierter Zeitanweisungen genutzt werden. Ein definiertes Voranschreiten des Prozeßsystems wird damit gesichert.
Die Kommunikation in Mehrrechnersystemen erfolgt ausschließlich über den Nachrichtenaustausch per SEND/WAIT-Primitiven. Genutzt wird eine inhomogene Interpretation des SEND-Rufparameters Zielprozeßnummer, von dem ein Warteteil einen Nachrichtenpuffer spezifiziert. Der Adressat einer Nachricht ist ein genau einmal im Gesamtsystem vorhandener Nachrichtenpuffer, von Typ und Größe gleich dem eines normalen Prozeßpuffers. Seine Lage und Verbindung zu einem bestimmten Prozeß werden in speziellen Konfigurationstabellen festgelegt.

Debugger DRC8000

Der Echtzeitdebugger DRTC8000 arbeitet in Verbindung mit dem Echtzeitbetriebssystemkern RTC8000 /5/ und dient zum Echtzeit-Test eines Anwenderprogrammpaketes in Verbindung mit den jeweiligen Programm-(Assembler-)listen. Unter anderem in Auswertung einiger Echtzeitdebugger für diverse Echtzeit-Betriebssystemkerne /10/ ... /12/ wurde der Debugger so entwickelt, daß ein einfaches und klares Konzept die unten beschriebene Teststrategie unterstützt.
Der Debugger ist ein Quasi-Anwenderprozeßpaket, welches höchstpriorisiert arbeitet. Das Besondere besteht darin, daß er über die Datenstruktur des RTC-Kerns informiert ist und mittels der beiden vom Kern angebotenen Globalreferenzen RTCKON, RTCDAT zu dieser unmittelbaren Zugang hat.
Des weiteren löst er die drei vom Kern geforderten Referenzen *Tentry, Texit, Setcou* auf. Mehr Bindungen bestehen nicht!
Ebenso wie der RTC-8000-Kern stellt der Debugger kaum Hardwareanforderungen. Sie reduzieren sich auf eine Zähler-Hardware sowie die Spezifika des Dialogkanals. Die zugehörigen Programmteile sind im Debuggermodul DEA.S im Quelltext gegeben, können vom Anwender gezielt editiert und im Bindelauf mit den übrigen Debuggerteilen zum vorgefertigten abgeschlossenen Debuggerobjektmodul zusammengefaßt werden.
Entsprechend den beiden Prozessorvarianten U8002 und U8001 gibt es zwei Versionen mit den korrespondierenden Objektmodulbezeichnungen DRTC2.OBJ bzw. DRTC1.OBJ (z. Z. nur die nichtsegmentierte Variante). Der Debuggerobjektmodul wird wie ein „normaler" Anwendermodul und im Gleichlauf mit diesen mit dem Kern gebunden.

Dipl.-Ing. Wolfgang Rehm (36) ist gelernter Elektromechaniker und erlangte 1975 das Diplom für Informationstechnik. Anschließend war er als Entwicklungsingenieur für UKW-Funktechnik im VEB Funkwerk Köpenick beschäftigt.
Seit 1981 ist er als wissenschaftlicher Assistent an der Technischen Universität Karl-Marx-Stadt (vormals Technische Hochschule Karl-Marx-Stadt) in Forschung und Lehre auf den Gebieten 16-Bit-Mikrorechentechnik, Echtzeit-Betriebssysteme und verteilte Betriebssysteme tätig.

Tafel 3 Debuggerkommandos (Auszug)

Go	class		
Stop	class		
Next	class		
BLende			
BS	adr		
BP	adr		
–K–			
Register	pnr		
Rx	pnr		
Display	aadr	len	
CLass	pnr	class	
PRty	pnr	prty	
HOld	pnr		
COntn	pnr		
STArt	pnr	mes1	mes2
STARTI	pnr	mes1	mes2
STOPI	pnr		
SENd	pnr	mes1	mes2
ASSign	iveknr		
–K–			
RESign			
EVEnt	iveknr		
–K–			
Time	auftrag		
Trace	triggerbedingung		
Test	adr		
REasm	anfadr	länge	
allg. Debuggerkommandos			

Hinweis: –K– unmittelbar folgendes Debuggerkommando (typ. Kernruf)

Teststrategie

Ausgenutzt wird die Fähigkeit des RTC-Kerns, Prozeßklassen (bzw. Prioritäten) dynamisch verwalten zu können. Der Debugger besitzt einen Prozeß, dem eine Sonderaufgabe für den Start bzw. die Unterbrechung des Anwendersystems zugewiesen ist. Nach dem Debuggeranlauf arbeitet dieser „Blendprozeß" ohne freiwillige Prozessorfreigabe (Warteschleife) auf einem hohen, über dem Anwendersystem liegenden Rang high-class und blendet alle niederpriorisierten Teile aus.
Mit Ausführung des Kommandos Go wird die Klasse dieses Prozesses auf den Wert low-class gesetzt, und ein entsprechend geringerer Teil des Anwendersystems wird ausgeblendet.
Das Kommando Stop setzt die „Blende", sprich Klasse des Blendprozesses, wieder auf den Wert high-class. Die Distanz zwischen den Blendwerten kann soweit eingegrenzt werden, daß quasi nur noch ein Prozeß getestet wird.
Mit Hilfe des Kommandos Blende können die aktuellen Werte von high-class und low-class angezeigt werden.
Die Grundstrategie sollte sein, daß schrittweise die getesteten höherpriorisierten Prozesse in den Hintergrund verlegt werden, was mit einem schrittweisen Erniedrigen des high-class-Wertes einhergeht (Bild 2).

Kommandos

Der Einzelschrittbetrieb mittels des Kommandos Next entspricht einem Go mit anschließend automatischem (hardware-aktivierten) Stop. Die NVI-Interruptebene wurde als Betriebssystem-Interruptebene vorgesehen. Diese steht dem Anwender nicht zur Verfügung. Da sich der Kern gegen diese Interrupts sperrt, wird er stets in einem Schritt durchlaufen. Die Schrittlogik verwendet den NVI (nicht den NMI!). Das Next-Kommando kann folglich ohne Einschränkungen verwendet werden.
Zum gezielten Test einzelner Prozesse ist es notwendig, daß ihre Beziehungen zum übrigen Prozeßsystem nachgebildet werden können. Folglich wurden Debugger-Kommandos vorgesehen, die zielgerichtete Kernrufe und Ereignisse auslösen.
Zwei Prüfpunkte können gesetzt werden. Mit dem Kommando Break-System BS wird ein Soft-Prüfpunkt auf eine Adresse gesetzt, welcher nach dem Erreichen quasi ein Stop-Kommando ausführt; es werden also Teile des Anwendersystems deaktiviert.
Mit dem Kommando Break-Prozeß BP wird ein Soft-Prüfpunkt auf eine Adresse gesetzt, der nach dem Erreichen eine geplante Kernaktion auslöst, welche eine mehr lokale als globale Bedeutung für das übrige Anwendersystem haben sollte.
Durch Ausführung des Register-Kommandos werden alle Register eines Prozesses sowie der komplette Prozeßstatus in Gestalt des Prozeßstatuswortes ausgegeben. So bedeutet zum Beispiel die Ausgabe *eS.VN. Mr blAw s...,* daß der Prozeß nichtsegmentiert (e) und im Systemmodus (S) arbeitet, daß der

vektorierte und nichtvektorierte Interrupt freigegeben sind (V), (N), daß sich eine Message im Nachrichtenpuffer befindet (M) und der Laufzustand des Prozesses „angehalten“ (A) ist.
Ein besonderes Merkmal des Debuggers ist, daß er in Zusammenarbeit mit dem Kern eine Trace-Funktion unterstützt. Jeder Kerneintritt und -austritt wird kodiert in einem Ringpuffer abgelegt. Damit können der Echtzeitlauf der Kernrufe und die damit möglicherweise verbundenen Prozeßumschaltungen für einen definierten Zeitabschnitt verfolgt werden. Das Setzen definierter Triggerbedingungen für den Echtzeitlauf gestattet das „Herausschneiden“ bestimmter Zeitabschnitte.
Einen Kurzüberblick über die wesentlichen Debuggerfunktionen zeigt Tafel 3.

Bestandteile

Der Debugger besteht aus 9 Prozessen, denen die Prozeßnummern 1 bis 9 zugewiesen sind. Neben den Terminalhandler-, den Interrupt-, Blend- und Kommandoprozessen wird ein „leerer“ Prozeß zur Verfügung gestellt, dem beliebige (Unter-)Programme des Anwendersystems zugewiesen werden können. Die aktuelle Fortführungsadresse dieses Testprozesses kann mit dem Kommando Test überschrieben und der Prozeß selbst gestartet werden. Die Klasse des Testprozesses liegt nach der Initialisierung direkt unterhalb derjenigen des Blendprozesses. Damit können beliebige Programmteile kurzzeitig höchstpriorisiert ausgeführt bzw. getestet werden.

Des weiteren stellt der Debugger einen Uhrprozeß bereit, dem verschiedenste Zeitaufträge übertragen werden können.

Schlußbemerkung

Mit den Grundfunktionen

- Prozeßsystem-Start/Stop
- Setzen von Prüfpunkten mit Teilsystemstop
- Setzen von Prüfpunkten mit Auslösung definierter Kernaktionen
- Schrittbetrieb eines definierten Teils des Prozeßsystems
- Emulation von Kernrufen und Ereignissen
- Echtzeit-Tracefunktion
- rückassemblierte Anzeige definierter Codebereiche

ermöglicht der vorgestellte Debugger einen wirkungsvollen Test eines Echtzeitprogrammsystems auf der Basis des RTC8000-Kerns. Die Kernimplementierung selbst wurde mit dem Debugger auf einer USS8000-Standardkonfiguration /13/ getestet. Allgemeine Leistungseigenschaften und Codeumfang (etwa 4 bis 9 KByte) des Debuggers stehen in ausgewogenem Verhältnis zum Kern.

✉ KONTAKT ☎

Technische Universität Karl-Marx-Stadt,
Sektion Informationstechnik,
Reichenhainerstr. 70, Karl-Marx-Stadt, 9022;
Tel. 5613246.

Literatur

/1/ Antonov, A.: Echtzeit-Betriebssystemkern für Mikrorechner. Nachrichtentechnik Elektronik 32 (1982) 11, S. 447–451
/2/ Fey, P.; Hübner, U.: Neue Unterstützungssoftware für die höhere Echtzeitsprache PLZ/RTC. Nachrichtentechnik Elektronik 35 (1985) 8, S. 310–312
/3/ Fey, P.; Antonov, A.: Real-Time Extension of a Higher Level Programming Language for Microcomputers. Microprocessing and Microprogramming 15 (1985), p. 191–194
/4/ Hübner, U.; Held, U.: Programmiersprache PLZ für 16-bit-Mikroprozessoren. Nachrichtentechnik Elektronik 36 (1986) 11, S. 408–410
/5/ Rehm, W.: Echtzeit-Betriebssystemkern RTC8000. Programmbeschreibung. Techn. Hochsch. Karl-Marx-Stadt, Sektion Informationstechnik 1986
/6/ Rehm, W.: Kerngenerator RTCCG8000. Programmbeschreibung. Techn. Hochsch. Karl-Marx-Stadt, Sektion Informationstechnik 1986
/7/ Rehm, W.: Echtzeitdebugger DRTC8000. Programmbeschreibung. Techn. Hochsch. Karl-Marx-Stadt, Sektion Informationstechnik 1986
/8/ Rieken, R.; Kriesten, R.; Jost, J.: Bildverarbeitungssystem mit 16-bit-Mikrorechner. BILD UND TON 37 (1984) 11, S. 337–340
/9/ Antonov, A.; Hübner, U.: Höhere Echtzeit-Programmiersprache PLZ/RTC V. 3.10. Einführung. Institut für Nachrichtentechnik Berlin, Dokumentation 1986
/10/ Anwendungsbeschreibung MRTS 700. VEB Numerik „Karl Marx“, Karl-Marx-Stadt; 1985
/11/ Echtzeit-Testsysteme RTC. DEBUG (U800). Dokumentation. Techn. Hochsch. Karl-Marx-Stadt 1984
/12/ ZRTS USER MANUAL, Zilog Inc. 1981
/13/ Rehm, W.: Universelles 16-Bit-System USS8000. Radio, Ferns., Elektron. 35 (1985) 5, S. 282–285

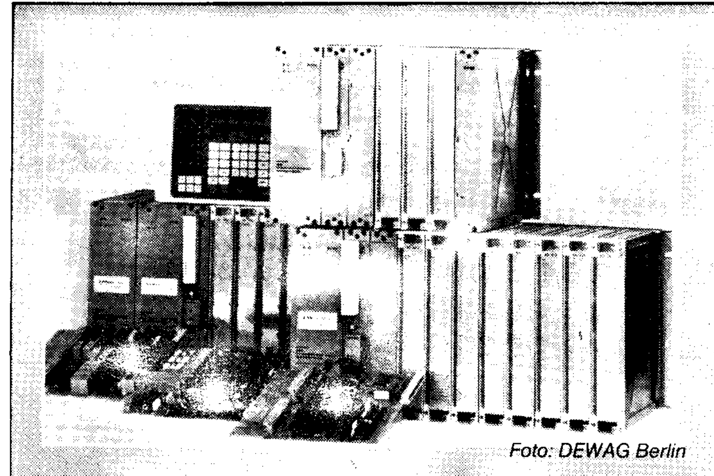


Foto: DEWAG Berlin

Gerätesystem zur prozeßnahen Automatisierung EAW electronic S 2000

Zur Lösung vielfältiger Automatisierungsaufgaben mit verfahrenstechnischem Charakter in der chemischen Industrie, der Kraftwerkstechnik, der Maschinen- und Aggregatautomatisierung, der Landwirtschaft, der Nahrungsgüterindustrie u. a. Industriezweigen produziert das Kombinat VEB Elektro-Apparate-Werke Berlin-Treptow das Gerätesystem zur prozeßnahen Automatisierung EAW electronic S2000. In den unterschiedlichsten Ausbauvarianten, ausgehend von einem für den Anwendungsfall optimal zugeschnittenen modularen Aufbau, bietet das System für den Anwender vorteilhafte Gebrauchswerte. Sie sind insbesondere in der einfachen Projektierung der Hardware, der Unterstützung des Projektierungsprozesses der Software über leistungsfähige Fachsprachen, der einfachen Anlagenerrichtung – bedingt durch kostengünstige Schraubklemmtechnik zum Anschluß der Prozeßsignale – sowie durch nicht erforderliche Schrankeinheiten zu ihrer Unterbringung zu sehen.
EAW electronic S2000 wird in Form des Mikrorechnerreglers EAW electronic S2000-R und als speicherprogrammierbare Steuerung EAW electronic S2000-S den Anwendern angeboten. Beide Einrichtungen sind speziell für ihre Einsatzbereiche in ihren Hard- und Softwareeigenschaften so ausgelegt, daß sie den Aufgabenstellungen optimal genügen. Die freie Programmierbarkeit der Einrichtungen sichert die Realisierung selbst komplizierter Aufgabenstellungen.
Die Softwarepakete PROMAR und PROLOG ermöglichen auf dem Programmiergerät P 8000, PC 1715 bzw. Bürocomputer A5120 einen durchgängigen CAD-Prozeß für die Anwenderprogrammentwicklung, die Inbetriebnahme der Einrichtungen sowie die Durchführung von Aufgaben der Simulation programmierter Strukturen und die Bemessung von Regelkreisen. Für die Kopplung mit Leit- bzw. übergeordneten Rechnern und zur Realisierung bildschirmorientierter Kommunikation stehen standardisierte serielle Schnittstellen zur Verfügung.
EAW electronic S2000 wird ab 4/1987 im Stammwerk des Kombinates Elektro-Apparate-Werke „Friedrich Ebert“ in Berlin Treptow produziert und noch 1987 in größerer Stückzahl den Anwendern in der DDR bereitgestellt.
Korner

Terminalanschluß an paketvermittelte Datennetze

Dr. Achim Hennecke, Franz Janitzek,
Norbert Klehn, Bernd Rieger
Akademie der Wissenschaften der DDR,
Institut für Informatik
und Rechentechnik Berlin

Der folgende Beitrag beschreibt den Anschluß von Terminals an ein öffentliches Datennetz mit einer X.25-Schnittstelle unter Verwendung einer Terminal-Kommunikationseinrichtung (TKE), die die Terminals von einem großen Teil der Kommunikationsfunktionen entlastet. Die Datenübertragung zwischen den Terminals und der TKE wird auf Basis dort standardmäßig vorhandener Schnittstellen und geeigneter Datenübertragungsverfahren realisiert. Die ersten auf den Terminals implementierten Anwendungen sind die Operatorkommunikation, der Fileübertragungsdienst und die entfernte Auftragsbearbeitung. Für eine effektive Realisierung der Kommunikationsfunktionen und der Anwendungen wurde eine einheitliche Basissoftware entwickelt.

1. Einführung

Paketvermittelte Datennetze mit einer X.25-Schnittstelle bilden eine weitverbreitete Grundlage als Informationsübertragungssysteme für den Aufbau von Rechnernetzen verschiedener Architektur und für verschiedene Anwendungen. Im besonderen gibt es eine wachsende Notwendigkeit, heterogene Rechnersysteme innerhalb eines Rechnernetzes zu verbinden. Das sollte in Übereinstimmung mit den Architekturprinzipien, die im OSI-Referenzmodell (OSI-RM) /1/ definiert sind, und mit den Standards, die für die Dienste und Protokolle der einzelnen Schichten entwickelt wurden, getan werden. Den Architekturprinzipien des OSI-RM folgend, muß auf allen Endsystemen die gesamte Hierarchie der sieben definierten Kommunikationsschichten zur Verfügung gestellt werden. Die Implementierung dieser Schichten ist durch das OSI-RM nicht festgelegt und kann deshalb von einem Endsystem zum anderen unterschiedlich sein. Sie hängt zum Beispiel von den Leistungsparametern der Rechnersysteme, die verbunden werden sollen (Hauptspeicher, Geschwindigkeit) und von der Implementierungsumgebung (Betriebssystem, Programmiersprachen usw.) ab.

2. Terminalanschluß

Ausgehend von den Erfahrungen bei der Entwicklung und Inbetriebnahme des Rechnernetzes DELTA /2/ wurden für den Anschluß von Endgeräten an das Datennetz Kommunikationseinrichtungen /3/ entwickelt. Während Arbeitsrechner mit Hilfe einer Arbeitsrechner-Kommunikationseinrichtung mit dem Datennetz verbunden sind, wird für den Anschluß der Terminals eine Terminal-Kommunikationseinrichtung (TKE) genutzt (Bild 1).

2.1. Struktur des Terminalanschlusses

Die TKE besteht aus den beiden logischen Teilen *Transportstation* (TS) und *Terminal-Anschlußsteuerung* (TAS).

Die TS realisiert die Verbindung der TKE zum Datennetz. Sie besteht aus den untersten vier Schichten des OSI-RM und stellt somit den Transportdienst bereit.

Die TAS ist verantwortlich für den Anschluß der Terminals an die TKE. Sie erfüllt keine Kommunikationsfunktionen im Sinne des OSI-RM, sondern realisiert eine transparente Übertragung der Transportschnittstellendateneinheiten zwischen der TKE und den angeschlossenen Terminals und stellt somit die Transportdienstschnittstelle auf dem Terminal bereit.

Eine entsprechende Partnerkomponente der TAS befindet sich auch auf den Terminals.

Die ersten auf den Terminals realisierten Anwendungen sind die für das Rechnernetz DELTA definierten Dienste zur Fileübertragung und Operatorkommunikation. Auf Basis des Fileübertragungsdienstes ist weiterhin die Realisierung der Entfernten Auftragsbearbeitung vorgesehen. Diese Anwendungen nutzen direkt die Schnittstellen des OSI-Transportdienstes (Bild 2).

2.2. Basissoftware

Als Voraussetzung zur Implementierung der Kommunikationsfunktionen wurde Basissoftware geschaffen. Sie wird sowohl für die Terminals als auch für die TKE genutzt.

Die Basissoftware besteht aus mehreren Komponenten (Bild 3):

- Die *Komponente zur Datenfernübertragung* realisiert eine zeichenorientierte asynchrone Datenübertragungsprozedur. Dazu wird die standardmäßig vorhandene V.24-Schnittstelle genutzt. In Abhängigkeit von der eingesetzten Datenübertragungseinrichtung sind Übertragungsgeschwindigkeiten von 1,2 ... 9,6 KBit/s möglich.
- Der *Prozeß-Supervisor* realisiert eine Prozeßsteuerung und stellt dem Anwender einen Satz von Routinen zur Programmierung von mehreren quasiparallel arbeitenden Prozessen auf einer 1-Prozessor-Maschine bereit. Die Kommunikation zwischen den Prozessen erfolgt ereignisgesteuert.
- Die Prozeßsteuerung macht es notwendig,

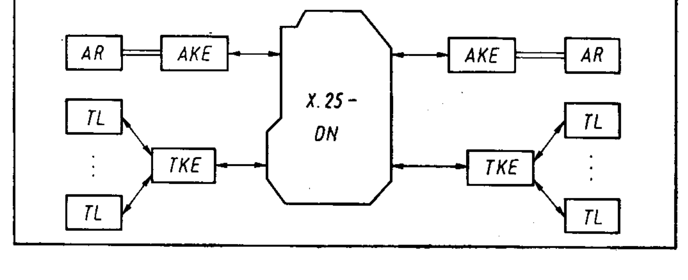


Bild 1 Rechnernetz mit Endsystemen, die über Kommunikationseinrichtungen mit dem Datennetz verbunden sind
AR – Arbeitsrechner
TL – Terminal
DN – Datennetz
AKE – Arbeitsrechner-Kommunikationseinrichtung
TKE – Terminal-Kommunikationseinrichtutung

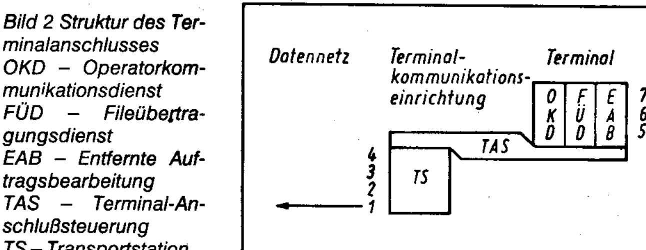


Bild 2 Struktur des Terminalanschlusses
OKD – Operatorkommunikationsdienst
FÜD – Fileübertragungsdienst
EAB – Entfernte Auftragsbearbeitung
TAS – Terminal-Anschlußsteuerung
TS – Transportstation

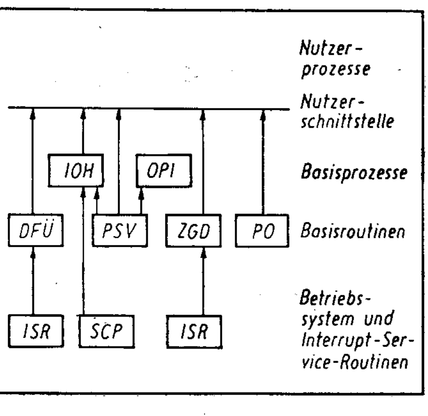


Bild 3 Struktur der Basissoftware
IOH – Ein-Ausgabebehandler
OPI – Bedienereingabe
DFÜ – Komponente zur Datenfernübertragung
PSV – Prozeßsupervisor
ZGD – Zeitgeberdienst
PO – Pufferorganisation
ISR – Interrupt-Service-Routine
SCP – Betriebssystem

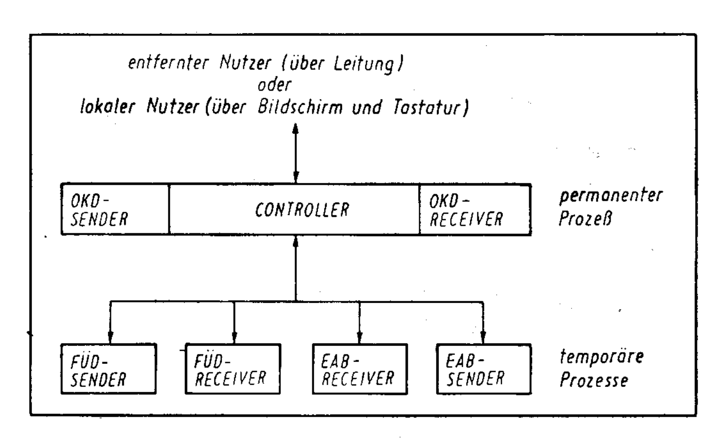


Bild 4 Struktur der Anwendungsdienste

Achim H[illegible]ecke (37) st[illegible]erte an der Elektrotechnischen Hochschule in Leningrad und promovierte 1983 an der AdW der DDR zum Dr.-Ing. Er widmete sich auf dem Gebiet der Rechnernetze Problemen des Transportdienstes. [illegible] ist Leiter einer [illegible]schungsgruppe, die sich mit dem Term[illegible]zugang zu öffentlichen Datennetzen beschäftigt.

Franz Janitzek (55) ist Diplom-Physiker. Seit 1972 arbeitet er auf dem Gebiet der Softwareentwicklung für Kleinrechner, [illegible]besondere im Rahmen der Entwicklung [illegible] Terminalsystems [illegible] Rechnernetzes DELTA. Er beschäftigt sich mit der Problematik der Rechnernetzanwendungen für Terminals.

Norbert Kiehn (32) erwarb 1981 ein Mathematik-Diplom an der Humboldt-Universität zu Berlin. Danach arbeit[illegible] der Entwicklung des [illegible]erminals[illegible] Rechnernetzes DEL[illegible] Gegen[illegible] sich Problem[illegible] der Dat[illegible] zwischen Mikrorechnern.

[illegible] erwarb 1973 an [illegible] Technischen Universität Dresden ein Ingenieur-Diplom für Informationsverarbeitung. Nachdem er an der Entwicklung des Terminalsystems des Rechnernetzes DELTA gearbeitet hatte, widmete er sich dem Entwurf und der Entwicklung von Basissoftware für Kommunikationsanwendungen für Mikrorechner.

E/A-Operationen (einschließlich Datenübertragung) zu überwachen und zu koordinieren. Diese Funktion erfüllt die Komponente *Ein-/Ausgabe-Behandler.* Dem Anwender wird eine einheitliche Schnittstelle zum Formulieren seiner E/A-Aufträge bereitgestellt. weiterhin stehen mehrere Routinen zur Bildschirmsteuerung und -ausgabe bereit.

- Die *Pufferorganisation* ermöglicht eine dynamische Speicherverwaltung und somit eine effektive Nutzung des Hauptspeichers.
- Der *Zeitgeberdienst* ist eine Komponente, die einen Time-Out-Service und eine Echtzeituhr realisiert. Diese Dienste werden dem Anwender ebenfalls über bestimmte Routinen zur Verfügung gestellt.
- Die Steuerung der Anwendersoftware kann kommandogesteuert erfolgen. Durch Betätigung einer Gesuchtaste können Kommandos eingegeben werden. Neben den internen Kommandos der Basissoftware (CLOCK, DIR, DUMP, ERA, HARDCOPY, REN, RESET, TRACE) kann der Anwender eigene Kommandos definieren. Eine entsprechende Schnittstelle stellt die Komponente *Bedienereingabe* bereit.

Somit kann diese Basissoftware als Standard-Implementierungsumgebung für die Realisierung von Netzanwendungen genutzt werden. Die Anwendungen können in den Programmiersprachen PASCAL oder Assembler erstellt werden. Der Hauptspeicherplatzbedarf der Basissoftware beträgt je nach Ausbaustufe 10 .. 15 KByte.

2.3. Realisierung

Die Lösung beruht auf *Gerätetechnik* aus DDR-Produktion. Als Terminals kommen Bürocomputer der Typen A 5120 und A 5130 sowie der Personalcomputer PC 1715 zum Einsatz. Für alle eingesetzten Terminals ist das Betriebssystem SCP verfügbar. Als TKE wird ein modifizierter Konzentrator K 8521 eingesetzt. An eine TKE können maximal 16 Terminals angeschlossen sein.

Die *Terminal-Anschlußsteuerung* (TAS) ist die Software-Komponente, die die Datenübertragung zwischen der TKE und den angeschlossenen Terminals übernimmt. Entsprechend den Funktionen besteht die TAS aus zwei logischen Teilen, aus der Übertragungskomponente und aus dem Schnittstellenbehandler. Die Übertragungskomponente realisiert die Datenübertragung mit Hilfe des von der Basissoftware bereitgestellten Übertragungsverfahrens und darüber hinaus weitere Fehlererkennungsmaßnahmen, um eine gesicherte Datenübertragung zu erreichen. Die Übertragungskomponente ist unabhängig von der Schnittstelle zwischen der Transport- und Sitzungsschicht des OSI-RM. Die Anpassung zwischen dieser Schnittstelle und der Übertragungskomponente ist Aufgabe des Schnittstellenbehandlers. Des weiteren ergreift er eine der Transportdienstschnittstelle entsprechende Maßnahme im Falle nichtbehebbarer Übertragungsfehler. Die TAS ist als Nutzerprozeß realisiert, wobei für jede Leitung ein Prozeß generiert wird.

Die ersten auf den Terminals realisierten Anwendungen sind die für das Rechnernetz DELTA definierten Dienste zur Fileübertragung (FÜD) und Operatorkommunikation (OKD).

Der Dienst OKD ermöglicht die Kommunikation zwischen beliebigen im Rechnernetz verbundenen Teilnehmern. Dabei wird eine am Terminal über Tastatur eingegebene Mitteilung zum Empfänger übertragen und dort auf dem Bildschirm angezeigt.

Der FÜD überträgt ein File von einer Diskette auf einen externen Datenträger des Empfängers. Ist der Empfänger ein Terminal, kann das File auf Diskette bzw. auf Drucker (bei druckbaren Files) ausgegeben werden. Der Fileinhalt ist für den FÜD transparent. Es besteht die Möglichkeit, eine Fileübertragung operatorgesteuert abzubrechen bzw. wieder fortzusetzen. Der FÜD kann gleichzeitig ein File senden und empfangen.

Auf Basis des FÜD ist weiterhin die Realisierung des Dienstes Entfernte Auftragsbearbeitung vorgesehen. Bei diesem Dienst wird das vom Terminal gesendete File beim Arbeitsrechner als Jobinput aufgefaßt und veranlaßt, daß der Joboutput dem entsprechenden Terminal zugestellt wird.

Die Anwendungen nutzen direkt die Schnittstelle des OSI-Transportdienstes. Die Schnittstelle zum Nutzer der Dienste wurde dialogorientiert gestaltet.

Die Implementierung der Anwendungen ist dadurch charakterisiert, daß sich jeder Dienst in eine Sende- und Empfangskomponente unterteilt (SENDER, RECEIVER). Jede Komponente ist als temporärer Prozeß realisiert, der durch eine zentrale Steuerkomponente (CONTROLLER) bei Bedarf generiert wird. Nach Beendigung der jeweiligen Funktion wird der entsprechende Prozeß gestoppt. Somit steht diese Ressource für andere Dienste wieder zur Verfügung. Eine Ausnahme hiervon bildet der OKD. Auf Grund der geringen Laufzeit beider OKD-Komponenten wurden sie als Prozeduren innerhalb des CONTROLLERS realisiert. Der CONTROLLER ist als permanenter Prozeß implementiert (Bild 4).

3. Ergebnisse

Der beschriebene Terminalanschluß unter Nutzung der TKE wurde bereits einer produktiven Nutzung zugeführt. Als Teilergebnis wurde ein BC/PC-Verbund auf Basis der TKE geschaffen, der unabhängig von der Kopplung an das Datennetz die Dienste OKD und FÜD zur Verfügung stellt. Er bildet somit ein allgemein nutzbares System zur Datenübertragung zwischen mehreren an eine TKE angeschlossenen BC bzw. PC. Dieser BC/PC-Verbund dient in einem Pilotprojekt als kommunikationstechnische Grundlage für die CAM-Lösung *Integrierte Produktions-/Transportkette Eisenerz,* mit der für den Hauptdispatcher des Eisenhüttenkombinats Ost Informationen aus dem Seehafen Rostock über den Erzzulauf als Entscheidungsgrundlage bereitgestellt werden.

Literatur

/1/ Informations Processing Systems – Open Systems Interconnection. Basis Reference Model ISO/IS7498

/2/ Carl, D.; u. a.: The Architecture of the Communication Services Hierarchy of the Computer Network DELTA and Experiences in its Implementation. Proc. COMNET '81, North Holland Amsterdam, Netherlands 1981

/3/ Hennecke, A.; Heymer, V.: OSI Communication Controller for Heterogeneous Computer Systems. Proc. COMNET '85, North Holland Amsterdam, Netherlands, 1986

✉ KONTAKT ☏

Akademie der Wissenschaften der DDR,
Institut für Informatik und Rechentechnik,
Dr. Achim Hennecke, Rudower Chaussee 5,
Berlin, 1199; Tel. 6743571

Termine

Jahrestagung der WGMA
Rechnergestützte Meßtechnik und Qualitätssicherung CAQ für CAD/CAM und CIM

WER? Wissenschaftlich-Technische Gesellschaft für Meß- und Automatisierungstechnik der Kammer der Technik
WANN? 12. bis 13. November 1987
WO? Leipzig
WAS?
Rechnergestützte
– fertigungstechnische
– verfahrenstechnische
– prüftechnische
– metrologische
– kostenoptimierte und
– programmtechnische
Qualitätssicherung
WIE? Teilnahmemeldungen schriftlich an: Präsidium der KDT, Sektretariat der WGMA, PSF 1315, Berlin, 1086

Müller

Universelle Nutzung des BASIC-Interpreters

Prof. Dr. Horst Völz, Berlin

Es werden der Arbeitsspeicher und die Routinen des BASIC-Interpreters von den KC-Rechnern in Tabellenform vorgestellt. Am Beispiel einer interaktiven, direkten Funktionseingabe wird gezeigt, wie man den Interpreter zusätzlich effektiv nutzen kann. Auf weitere Möglichkeiten wird hingewiesen.
Die Subroutinen des BASIC-Interpreters von den KC-Rechnern wurden bereits in MP 6/87, S. 182 veröffentlicht.

Einleitung

Die Kleinrechner KC 85/1, KC 85/2, KC 85/3 und KC 87 besitzen einen Interpreterkern von 8 KBytes, der identisch ist. Die bei den KC 85/1 und KC 85/2 verwendete Bandversion ist dabei allerdings additiv verschoben. Während die ROM-Version des Kerns von C000 bis DFFF liegt, wird die Bandversion von 200 bis 2A00 abgelegt. Sie enthält außerdem bereits Teile, die über den Kern hinausgehen.

```
20 DEFFNA(X)=X:'::::::::::::::::::::::::::::::::::::::::::::::::::::::
30 INPUT"F(X) =";A$: CALL*4DA
40 FOR X=1 TO 10
50 PRINT X,FNA(X)
60 NEXT
```

Bild 1 BASIC-Programm

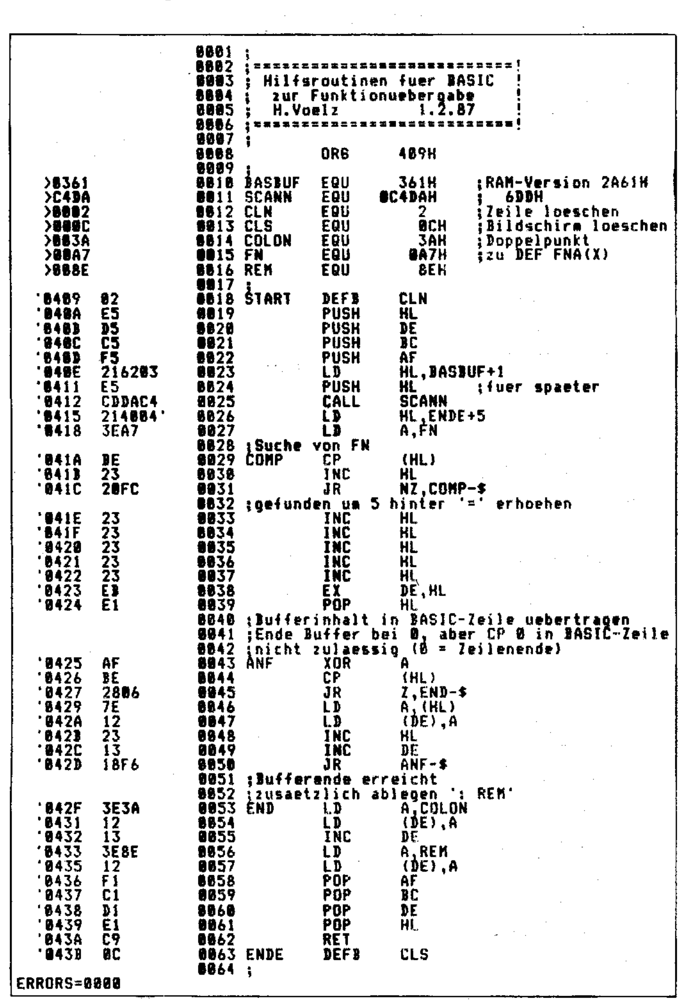
Diese Teile sind bei den Rechnern verschieden. Hier wird nur die Ergänzung bei der ROM-Version des KC 85/2 bzw. des KC 85/3 behandelt. Bei einiger Erfahrung mit dem Interpreter und bei guten Kenntnissen in Maschinensprache sind diese Angaben aber leicht auf die anderen Fälle übertragbar. Es sei erwähnt, daß für den Z 1013 die hier behandelte Bandversion ebenfalls verfügbar ist. Schließlich sind die meisten BASIC-Versionen recht ähnlich zu der hier vorgestellten. Daher können viele Angaben nahezu universell in BASIC genutzt werden.

Arbeitsspeicher

Der Arbeitsspeicher des ROM-Interpreters liegt von 300 bis 3FF (2A00 bis 2B00). Bis 363 werden die Speicherzellen beim Start von BASIC mit bestimmten (default-) Werten belegt. Sie sind in der Tabelle konkret aufgelistet. Die dann folgenden Werte sind erst beim Ablauf eines Programms bedeutsam. Auf alle Zellen kann man bei Bedarf zurückgreifen. Einige Befehle des BASIC tun dies direkt, z. B.: LINES (344), NULL (340), USR (303). Mit der Adresse 35F kann man z. B. den Start für ein BASIC-Programm verlagern. Dies ist eine Methode, um mehrere Programme, die im RAM geladen sind, unabhängig lauffähig zu halten. Außerdem kann hiermit freier Speicherraum für Maschinenprogramme realisiert werden. Dabei ist aber zu beachten, daß die Zelle bei Neustart immer wieder mit 401H belegt wird. Die weiteren Fakten sind aus der Tabelle direkt erkennbar.

Teilroutinen

Die Beschreibung der vielfältigen Teilroutinen wäre sehr umfangreich. Deshalb sei nur auf einige, spezielle eingegangen. Die Routine bei C4DA wird im nächsten Abschnitt noch genauer erklärt. Ab C986 steht eine Routine, welche einen Text mit ASCII-Zeichen in hexadezimale Zahlen umwandelt. Hierfür könnte z. B. mit INPUT die Folge „3509" der Zeichenkette A$ zugewiesen werden. Dann liegt dieser Text auch im Eingabepuffer ab 362. Setzt man nun HL auf diesen Wert und ruft dann mit CD 86 C9 diese Routine auf, so kann man danach in DE den HEX-Wert dieser Zahl nutzen. Ähnliche Routinen existieren zwar auch in den Betriebssystemen und könnten von dort genutzt werden. Anders ist es aber bereits mit der Arithmetik. Sie ist ab D45E vorhanden und erlaubt eine generelle Nutzung auch in Programmen, die nicht in BASIC geschrieben sind. Dadurch kann dann z. B. die nicht immer glücklich formatierte Zahlenausgabe umgangen werden. Außerdem lassen sich viele BASIC-Organisationen vermeiden. Dadurch wird die Arithmetik dann schneller, als sie im vollständigen BASIC ist. Auch eine Arithmetik mit komplexen Zahlen läßt sich so definieren. Eine besonders nützliche Verwendung von Routinen aus dem BASIC-Interpreter be-



```
                    0001 ;
                    0002 ;===========================!
                    0003 ; Hilfsroutinen fuer BASIC  !
                    0004 ;  zur Funktionuebergabe    !
                    0005 ;  H.Voelz          1.2.87  !
                    0006 ;===========================!
                    0007 ;
                    0008           ORG      409H
                    0009 ;
 >0361              0010 BASBUF   EQU      361H     ;RAM-Version 2A61H
 >C4DA              0011 SCANN    EQU     0C4DAH    ;  6DDH
 >0002              0012 CLN      EQU        2      ;Zeile loeschen
 >000C              0013 CLS      EQU      0CH      ;Bildschirm loeschen
 >003A              0014 COLON    EQU      3AH      ;Doppelpunkt
 >00A7              0015 FN       EQU     0A7H      ;zu DEF FNA(X)
 >008E              0016 REM      EQU      8EH
                    0017 ;
'0409   02          0018 START    DEFB     CLN
'040A   E5          0019          PUSH     HL
'040B   D5          0020          PUSH     DE
'040C   C5          0021          PUSH     BC
'040D   F5          0022          PUSH     AF
'040E   216203      0023          LD       HL,BASBUF+1
'0411   E5          0024          PUSH     HL        ;fuer spaeter
'0412   CDDAC4      0025          CALL     SCANN
'0415   214004'     0026          LD       HL,ENDE+5
'0418   3EA7        0027          LD       A,FN
                    0028 ;Suche von FN
'041A   BE          0029 COMP     CP       (HL)
'041B   23          0030          INC      HL
'041C   20FC        0031          JR       NZ,COMP-$
                    0032 ;gefunden um 5 hinter '=' erhoehen
'041E   23          0033          INC      HL
'041F   23          0034          INC      HL
'0420   23          0035          INC      HL
'0421   23          0036          INC      HL
'0422   23          0037          INC      HL
'0423   EB          0038          EX       DE,HL
'0424   E1          0039          POP      HL
                    0040 ;Bufferinhalt in BASIC-Zeile uebertragen
                    0041 ;Ende Buffer bei 0, aber CP 0 in BASIC-Zeile
                    0042 ;nicht zulaessig (0 = Zeilenende)
'0425   AF          0043 ANF      XOR      A
'0426   BE          0044          CP       (HL)
'0427   2806        0045          JR       Z,END-$
'0429   7E          0046          LD       A,(HL)
'042A   12          0047          LD       (DE),A
'042B   23          0048          INC      HL
'042C   13          0049          INC      DE
'042D   18F6        0050          JR       ANF-$
                    0051 ;Bufferende erreicht
                    0052 ;zusaetzlich ablegen ': REM'
'042F   3E3A        0053 END      LD       A,COLON
'0431   12          0054          LD       (DE),A
'0432   13          0055          INC      DE
'0433   3E8E        0056          LD       A,REM
'0435   12          0057          LD       (DE),A
'0436   F1          0058          POP      AF
'0437   C1          0059          POP      BC
'0438   D1          0060          POP      DE
'0439   E1          0061          POP      HL
'043A   C9          0062          RET
'043B   0C          0063 ENDE     DEFB     CLS
                    0064 ;
ERRORS=0000
```

Bild 2 Assemblerausdruck

KC 85/3-BASIC-ARBEITSZELLEN

ADR	BYTE	BESCHREIBUNG
300	3	C369C0 Sprungbefehl Warmstart (REBASIC)
303	3	C367C9 Sprungbefehl in die USR(X)-Funktion. Der Anwender legt die Startadresse der MC- Routine auf 0304H ab DOKE 772,AD
306	1	00 Toradresse Variable
307	1	00 E/A -Flag
308	1	00 OUT-Index
309	1	00 IN-Index
30A	1	00 TRACE -Flag
30B	14	Schnelle SBC-Routine für Gleitkommadivision. Variablen auf Adr: 30CH 310H 314H 317H
319	3	000000 RND -Variablen
31C	32	Daten für die RND-Funktion (Pseudo-Zufallszahlen)
33C	4	52C74F80 Pseudo-Zufallszahl für RND(0) = .811635
340	1	08 Anz.der Dummy-Zeichen Veränderug in BASIC: NULL X 0<=X<255
341	1	FF Ausgabezeilenlänge (255)
342	1	1B (Anzahl der darstellbaren Formate pro Zeile-1) Formatlänge(13)+1 =27
343	1	00 Ausgabeunterdrückung wenn <>0
344	2	0A00 Zeilenanzahl für LIST. Normal:10 (LINES X)
346		0A00 Copy von 344
348	2	00 Prüfsumme für CLOAD* und CSAVE*
34A	3	C3AEC5 Eingaberoutine für Daten
34D	1	00 AUTO-Flag
34E	2	0000 Akt. Zeilennummer
350	2	0000 Zeilennr. -Abstand
352	2	0000 (NANF)
354	2	0000 (DISTAN)
356	2	6504 Ram-Mindestgröße (PRAM+100)
358	2	FEFF Akt. Zeilennummer (65534)
35A	3	0000C9 WINJP
35D	1	00 DATBYT
35E	1	00 DATFLG (List/Editierschutz) wenn <>0
35F	2	0104 Programmadresse (401H)
361	74	000000 Eingabepufferanfang init.361-363
3AB	1	... Eingabepufferende
3AC	1	Cursorposition
3AD	1	Suche Variable (0) oder füge sie ein
3AE	1	Datentyp
3AF	1	Datastmnt. Flag
3B0	1	Memory Size
3B2	2	Adresse der nächsten freien Stelle in der Stringadresstabelle
3B4	11	Stringadresstabelle
3BF	1	Ende der Tabelle
3C0	4	Länge und Adresse des aktuellen Strings
3C4	2	Zeiger für String
3C6	2	Index des zuletzt abgearbeiteten Bytes
3C8	2	Adresse des Programmzeigers während der Suche nach einer vorherigen FOR - Anweisung
3CA	2	Zeilennr. des zuletzt gelesenen DATA -Statements
3CC	1	FOR - Flag
3CD	1	Inputphase - Flag
3CE	1	RD/WR - Flag
3CF	2	Zeilennr./Adresse des akt. Statements
3D1	2	Adresse des nächsten Elements
3D3	2	Letzte Zeilennr. bei END / STOP
3D5	2	Adresse des letzten Bytes bei ERROR
3D7	2	Adresse der Liste der Einfach-und String-variablen
3D9	2	Adresse der Liste der Feldvariablen
3DB	2	Erste freie Adresse hinter den Listen
3DD	2	Akt. Adresse im DATA-Statement während READ
3DF	2	Adresse des FN - Statements
3E1	4	Arithmetikregister 0
3E5	4	Arithmetikregister 1
3E9	1	Vorzeichen der Operation. (SGN)
3EA	13	PRINT - Puffer
3F7	4	Arithmetikregister 2
3FB	1	Listen-Flag (Status) 0=Grundliste 1=Erweiterung
3FC	1	Erweiterungsflag 0= Erw. nicht vorhanden
3FD	1	Print-Erw. Flag 0= ohne Print-Erweit.
3FE	1	FARBBYTE

ENDE

Bild 3 Arbeitszellen des Interpreters

steht darin, zusätzliche Routinen, z. B. über CALL oder USR, sich selbst zu definieren. So ist zuweilen ein GOTO wertvoll, welches nicht Zeilennummern enthält, sondern einen arithmetischen Ausdruck verwendet. Ein solcher Befehl ist mächtiger als ON GOTO.
Noch eine Möglichkeit besteht darin, das in größeren Interpretern vorhandene ON ERROR GOTO nachzubilden. Hierzu ist in 303H die entsprechende Routine zu notieren und weiter sind die Fehlnummern auszuwerten. Sie sind nach C33E beschrieben.

Interaktive Funktionseingabe

Ein großer Nachteil von BASIC ist es, daß spezielle Funktionen stets programmiert werden müssen. Für Anwendungsprogramme, die auch „Rechnerlaien" verwenden sollen, ist es geradezu notwendig, eine Funktion im Programmablauf – ähnlich wie eine Zahl oder ein String – eingeben zu können. Das soll hier etwas ausführlicher beschrieben werden. Es wird je ein kurzes BASIC- und Maschinen-Programm verwendet (Bilder 1 und 2).
Das Maschinenprogramm wird in einer Zeile 10 unmittelbar nach REM abgelegt. Das geschieht am besten, indem man zunächst das BASIC-Programm – wie abgedruckt – eingibt. Zusätzlich wird eine Zeile 10 erzeugt:

```
10 !::::::::::::::::::::::::::::::::::::::
```

und zwar mit möglichst vielen Doppelpunkten. Auch in der Zeile 20 sind die maximalen Doppelpunkte zu erzeugen!
Danach geht man in den Betriebsmode zurück und gibt den Maschinencode entsprechend den Adressen ein. Geht man mit REBASIC nun zurück, so ist das Programm im wesentlichen als Einheit fertig. Bevor es genutzt wird, sollte es auf alle Fälle gespeichert werden.

Das Programm funktioniert wie folgt:
Bei RUN wird mit INPUT die eingegebene Funktion als String A$ zugewiesen und zugleich im BASIC-Puffer abgelegt. Dann wird die Maschinenroutine ab 40AH aufgerufen. HL wird dort auf Bufferbeginn gesetzt. Mit der BASIC-Routine SCANN ab 412 (C4DA) wird der ASCII-Text in die interne Darstellung des Interpreters umgewandelt. Im nächsten Teil des Maschinenprogramms ab 418 wird nun im BASIC-Programm jene Stelle gesucht, wo das Token FN (A7) steht. Ist diese Adresse gefunden, so wird um 5 erhöht. Dort steht in Zeile 20 das X nach =. Hier wird der interne Text der Funktion abgelegt (ab 425). Das Ende ist im BASIC-Puffer mit 0 gekennzeichnet. Hierauf muß also beim Ablegen getestet werden. Dies kann aber nicht über CP 0 erfolgen. Denn im Maschinencode einer REM-Zeile darf keine 0 vorkommen. Sie würde der Interpreter als Zeilenende verstehen und dadurch das Maschinenprogramm zerstören.
Nachdem die intern dargestellte Funktion in der Zeile 20 abgelegt ist, muß der Text dort noch gesichert werden. Er wird mit einem Doppelpunkt und REM abgeschlossen (42F). Die Zeichen 2 (409) und 0C (43B) sind Hilfszeichen, die das störende Auflisten der REM-Zeile 10 mindern.
Von der Wirkung der Eingabe einer Funktion kann man sich schnell durch BREAK und LIST 20 überzeugen.
Die Zeilen 10 und 20 sollten bei der Arbeit mit einem derartigen Programm nicht editiert werden. Sie würden dann nämlich zerstört bzw. in der Länge verkürzt.
Die Doppelpunkte in 20 (zunächst auch 10) sind Platzfreihalter für die Eingaben. Stehen dort erst einmal BASIC-Tokens, so ist die Darstellung auf der Zeile länger! Deshalb führt dann Editieren zum Verkürzen der Zeile. Die Zeilen 40 bis 60 im BASIC-Programm dienen nur der Demonstration. Sie können für Anwendungen durch beliebig andere BASIC-Zeilen ersetzt werden. Dieses Programm mit der Maschinenzeile 10 ist also Ausgangspunkt für die Programmentwicklung.
Übrigens läßt sich mit dieser Methode auch nach anderen Tokens suchen. Der Autor hat Programme geschrieben, in denen mehrere unterschiedliche Funktionen interaktiv an verschiedenen Stellen abgelegt werden.

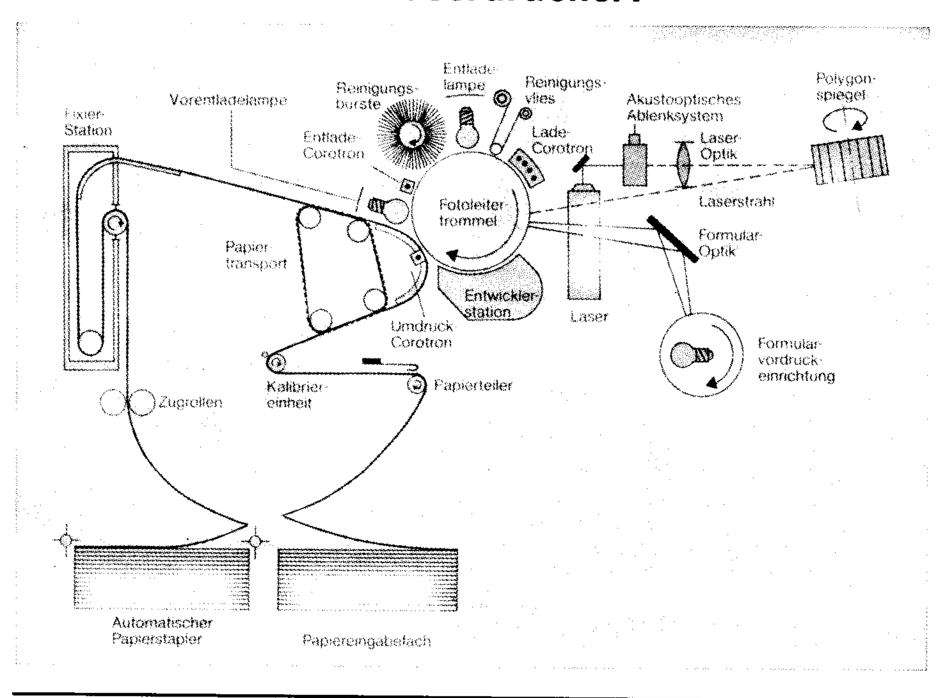
Bemerkungen

Für die Aufstellungen standen ein Ausdruck des ursprünglichen Interpreters vom Forschungszentrum für Tierproduktion Dummerstorf – Rostock, eines Microsoft-Interpreters und Hinweise aus der Literatur zur Verfügung. Es konnten nicht alle ermittelten Routinen erprobt werden. Daher besteht die Möglichkeit, daß einige Angaben Fehler enthalten.
Des Umfanges wegen, konnten zu den einzelnen Routinen auch nicht alle Bedingungen und die Ein-, Austrittsparameter angegeben werden. Der Autor ist interessiert, entsprechende Hinweise zu erhalten.

✉ KONTAKT ☏

Akademie der Wissenschaften der DDR,
Zentralinstitut für Kybernetik und
Informationsprozesse, Prof. H. Völz,
Kurstraße 33, Berlin, 1086

Wie funktioniert ein Laserdrucker?



Der Laserdrucker arbeitet mit Lasertechnologie und elektrofotografischem Druckprinzip. Eine rotierende, mit einem Fotohalbleiter beschichtete Trommel, die sog. Fotoleitertrommel, wird bei jeder Umdrehung elektrostatisch aufgeladen. Der Laserstrahl entlädt die Halbleiterschicht partiell und zeichnet so die zu druckende Information auf die Fotoleitertrommel; mehrere tausend Zeichen pro Minute. Da die Systeme mit einer hohen Auflösung arbeiten, lassen sich Zeichen in verschiedenen Größen und Schriftarten und auch Grafiken darstellen. Der Laserstrahl wird von einem akustooptischen Ablenker gesteuert, die horizontale Veränderung übernimmt ein Polygonspiegel, der sich 3000 mal in der Minutel dreht. Beim Entwickeln nehmen diejenigen Stellen der Fotoleitertrommel Toner auf, die der Laserstrahl vorher entladen hat. In der Umdruckstation wird das auf der Trommel entstandene Druckbild auf das Papier übertragen und anschließend fixiert.
Nach dem Druck entlädt eine Lichtquelle die Halbleiterschicht, eine rotierende Bürste entfernt den verbliebenen Toner. Somit kann der Prozeß wieder von vorn beginnen. Die Grafik zeigt das Funktionsprinzip.

Bild: Siemens-Information

Programmsystem CZPLOT

Manfred Berner, Dietmar Fürste
Forschungszentrum des Kombinates
VEB Carl Zeiss JENA

Der Beitrag informiert über das modular aufgebaute, weitestgehend hardwareunabhängige Grafiksoftwarepaket CZPLOT, in dem für den Bürocomputer A 5120 mit dem Betriebssystem UDOS oder für kompatible Systeme grafische Grundroutinen realisiert werden. Hinweise auf die zur Nutzung erforderliche Systemumgebung, die Anwendungsmöglichkeiten und Grenzen sollen dem Leser die Entscheidung über die vom Kombinat VEB Carl Zeiss JENA angebotene, kostenpflichtige Nachnutzung ermöglichen.

Zielstellung

Bei vielen innerbetrieblichen Prozessen, aber auch in Erzeugnissen des wissenschaftlichen Gerätebaus, hat die Mikrorechentechnik inzwischen einen festen Platz. Auf dem Gebiet der grafischen Datenverarbeitung spielen Mikrorechner, vor allem die 8-Bit-Systeme der U880-Familie, zur Zeit ebenfalls eine wesentliche Rolle. Als grafisches Ausgabegerät hat der Digitalzeichentisch DZT 90 × 120/RS des Kombinates VEB Carl Zeiss JENA /1/ eine weite Verbreitung gefunden. Bei vielen Anwendern ist der Wunsch entstanden, ihre in der Eingangssprache des DZT gespeicherten Bilder auch auf anderen Ausgabegeräten, wie z.B. grafikfähigen Druckern, Plottern oder einem Grafik-Display, darzustellen. Dies ist vor allem dann der Fall, wenn bei der Ausgabe von Zwischenprodukten zugunsten einer höheren Geschwindigkeit eine Einbuße an Genauigkeit in Kauf genommen wird.

Aus diesem Grunde sollte ein Programmsystem entwickelt werden, das die recht komfortablen Grafikbefehle des DZT interpretierend in einfache Polyline-Kommandos übersetzt, deren Verarbeitung bei jedem grafischen Ausgabegerät als Mindestniveau vorausgesetzt werden kann.

Damit wird es möglich, jedes beliebige grafische Ausgabegerät mit einfachen Treiberprogrammen über vereinheitlichte Schnittstellen in das System einzubinden. Ein derartiger Treiber wurde für das Robotron-Rastersichtgerät (RSG) K 8917 /2/ erarbeitet. Er

Befehl	Bedeutung	Wirkung
P x y	Position	Positionieren auf den Koordinatenpunkt (x,y)
S x y	STRAIGHT LINE	Geradlinige Verbindung von akt. Pos. nach (x,y)
C r	CIRCLE	Vollkreis mit Radius r um akt. Position
A x y r rm rb	ARC	Bogen mit Radius r, Mittelpunktlage rm, Bogenrichtung rb von akt. Pos. nach (x,y)
SB name	SYMBOL BEGIN	Start Symbol Generieren
SE	SYMBOL END	Ende Symbol Generieren
LB name	LINE BEGIN	Start Linienart Generieren
LE	LINE END	Ende Linienart Generieren
SY n	SYMBOL	Zeichnet generiertes oder Standard-Symbol
SY 'string'	SYMBOL 'string'	Schreibt Zeichenkette ab aktueller Position
LI n	LINE	Auswahl von generierter oder Standard-Linienart
LI 'string'	LINE 'string'	Zeichenkette als Linienart
I n	INSTRUMENT	Liniendicke im K8917-Bild
Z x y	ZERO	Setzt relativen Koordinaten-Nullpunkt auf (x,y)
EL f	ENLARGEMENT	Maßstabsfaktor f für alle Folgebefehle (ZOOM)
SZ mx my	SIZE	getrennte Maßstabsfaktoren für X- und Y-Richtung
SS m	SIZE SYMBOL LINE	Maßstabsfaktor für Symbole und Linienarten
RT a	ROTATION	Drehung des Koordinaten-Systems um a (rad)
AG a	ANGLE	Drehung des Koord.-Systems, wirkt nicht auf SY-Befehle
AS a	ANGLE SYMBOL	Dreht nur Symbole um die aktuelle Position um a (rad)
RF x y a	REFLEXION	Koordinaten, Zeichenketten u. Symbole an der Geraden durch (x,y) mit Steigung m spiegeln
VP x1 y1 x2 y2	VIEWPORT	Zeichnen nur innerhalb des gewählten Ausschnittes
RD n	READ	akt. Position oder Fehlerstatus auf Konsole ausgeben
RS n	RESET	Standardwerte einsetzen mit oder ohne Bildlöschen

Tafel 1 Eingangssprache für den Grafikkern CZGRAF

UDOS oder CZ-SDOS Executive — IIYI — CZPLOT Steuermodul — IIYI — Bild-Datei 1 (ASCII) — Anwender-Ebene
Schnittstelle 1: IIYI
CZGRAF Grafikkern — Grafik-Verarbeitungs-Ebene
Schnittstelle 2: IIYI
$K8917 $BA8917 — $Grafile — Handler-Ebene
Schnittstelle 3: IIFSSI — IIYI
RSG K 8917 — ZDOS — Workstation-Ebene
Schnittstelle 4: IIFSSI — IIYI
Drucker 1157/269 — Bild-Datei 2 (ASCII)
IY = Request-Vektor

Bild 1 Das Programmsystem CZPLOT und der Datenfluß

Befehl	Bedeutung	Wirkung
P x y	Position	Positionieren auf den Koordinatenpunkt (x,y)
S x y	STRAIGHT LINE	Geradlinige Verbindung von akt. Pos. nach (x,y)
LI n	LINE	Auswahl von generierter oder Standard-Linienart
I n	INSTRUMENT	Liniendicke im K8917-Bild
RD n	READ	akt. Position oder Fehlerstatus auf Konsole ausgeben
RS n	RESET	Standardwerte einsetzen mit oder ohne Bildlöschen

Zu den Ausgangsbefehlen des Grafikkerns CZGRAF rechnen außerdem **auch** alle nicht in Tafel 1 enthaltenen und ihm somit nicht bekannten Kommandos, die unverändert an das nachgeordnete Programm weitergegeben werden.

Tafel 2 Ausgangsbefehle des Grafikkerns CZGRAF

Befehl	Bedeutung	Wirkung
P x y	Position	Positionieren auf den Koordinatenpunkt (x,y)
S x y	STRAIGHT LINE	Geradlinige Verbindung von akt. Pos. nach (x,y)
C r	CIRCLE	Vollkreis mit Radius r um akt. Position
A x y r rm	ARC	Kreisbogen. Radius r, Mittelpkt.-Lage rm entgeg. Uhrzeigersinn von akt. Pos. nach (x,y)
SB name	SYMBOL BEGIN	Stoppt Grafikausgabe
SE	SYMBOL END	Grafikausgabe fortsetzen
LB name	LINE BEGIN	Stoppt Grafikausgabe
LE	LINE END	Grafikausgabe fortsetzen
SY n	SYMBOL	Standardsymbol zeichnen
SY 'string'	SYMBOL 'string'	Schreibt Zeichenkette ab aktueller Position
LI n	LINE	Wählt Standard-Linienart
LI 'string'	LINE 'string'	Zeichenkette als Linienart
I n	INSTRUMENT	Liniendicke im K8917-Bild
RD n	READ	akt. Pos. od. Fehlerstatus für Abfrage bereitstellen
RS 0	RESET	Bildlöschen u. CLEAR RSG
WD n b	DEFERAL STATE	Verzögerungsmode für RSG
WW x1 x2 y1 y2	WORKSTATION WINDOW	Setzt WINDOW im RSG
WV x1 x2 y1 y2	WORKSTATION VIEWPORT	Setzt VIEWPORT im RSG

Hinweis:
Die letzten drei Befehle sind erforderlich, um beim Initialisieren des RSG K8917 den erforderlichen Verzögerungsmodus und definierte WINDOW- und VIEWPORT-Parameter zu setzen.

Tafel 3 Eingangssprache für Handler $K8917 und $BA8917

Manfred Berner (39) studierte von 1966 bis 1970 an der Technischen Universität Dresden, Sektion Informationstechnik, im Bereich Technische Kybernetik und Automatisierungstechnik. Anschließend war er in verschiedenen Industriezweigen auf dem Gebiet der Einsatzvorbereitung und Programmierung von Prozeßrechnern tätig. Seit 1980 arbeitet er im Forschungszentrum des VEB Carl Zeiss JENA als Gruppenleiter im Bereich für Gerätesoftwareentwicklung.

Dietmar Fürste (47) ist Funkmechaniker und arbeitete nach dem Besuch der Ingenieurschule für Feinwerktechnik Jena ab 1961 zunächst als Kundendienstingenieur, ab 1966 im Labor für Feinmeßgeräte des VEB Carl Zeiss JENA. Nach einem Fernstudium an der Sektion Technologie der FSU Jena war er ab 1975 im Labor für Geräte der Medizintechnik eingesetzt: Mitarbeit bzw. Themenleitung bei verschiedenen Geräteentwicklungen umfaßten Aufgaben der Feinmechanik, der Optik und Elektronik und führten zu mehreren Patenten.
Nach Selbststudium und Spezialisierung auf die Hard- und Software von Mikrorechnern seit 1984 in einem Bereich für Software-Entwicklung tätig, seit 1986 als Systemprogrammierer.

kann sowohl als Komponente des gesamten Systems als auch im Solobetrieb verwendet werden.

Ergebnis

Es liegt ein modular aufgebautes, in U880-Assembler geschriebenes Programm vor, das aus mehreren *Ebenen* besteht (vgl. Bild 1):

- Anwenderebene
- Grafische Verarbeitungsebene
- Treiberebene (Handlerebene)
- Workstation- (Geräte-) Ebene.

Als Schnittstelle für den Datenaustausch zwischen den einzelnen Ebenen wird grundsätzlich der für die Betriebssysteme UDOS und CZ-SDOS typische Request-Vektor verwendet.
In der Anwenderebene realisiert der CZPLOT-Steuermodul folgende *Aufgaben:*

- Lesen von Bilddateien (ASCII-Files mit Grafikbefehlen) und Weitergabe an die untergeordnete Ebene
- Realisierung der Eingabe von Grafik-Einzelbefehlen auf der Rechnerkonsole und Weitergabe an die untergeordnete Ebene.

Dieses Programm kann durch beliebige andere Anwenderprogramme ersetzt werden.
Der Grafikkern CZGRAF übersetzt die in Tafel 1 beschriebene Eingangssprache, die im wesentlichen dem Befehlssatz des DZT /3/ entspricht und liefert als Ergebnis nur die einfachen grafischen Befehle, die in Tafel 2 beschrieben sind. Erhält das Programm CZGRAF andere Befehle als in Tafel 1 vereinbart, so werden diese unverändert an die nächst niedrigere Ebene weitergegeben. Dadurch ist es möglich, spezifische Eigenschaften einzelner Geräte anzusprechen, die nicht im Umfang des Befehlssatzes des CZGRAF enthalten sind, z. B. einige Befehle, die für die Initialisierung des RSG und das Setzen von WINDOW und VIEWPORT notwendig sind.
Auf Handlerebene können vom Anwender leicht gerätespezifische Programme erstellt werden, die die Verbindung zu den konkreten Ausgabegeräten herstellen. Diese Handler müssen die in Tafel 2 beschriebenen Polyline- und Steuerbefehle in geräteabhängige, hardwarenahe Ausgabebefehle umsetzen. Für das Rastersichtgerät K 8917 steht der Treiber $K8917 zur Verfügung, der den in Tafel 3 beschriebenen Sprachumfang verarbeitet. Damit bieten sich folgende *Anwendungsmöglichkeiten:*

- Einbindung von $K8917 in das Programmsystem CZPLOT
- Nutzung aller Ausgabefunktionen des RSG auch im Solobetrieb (ohne CZPLOT).

Die konsequente Verwendung der für das Betriebssystem UDOS typischen Schnittstellen bringt folgende *Vorteile:*

- Aufruf der Handler über Betriebssystem-Kommandos möglich, z. B. MOVE und COPY
- Einfache Einbindung der Handler in problemorientierte Sprache möglich, z. B. PASCAL, FORTRAN, BASIC oder FORTH.

Ein Beispiel für die Einbindung des Handlers $K8917 in BASIC zeigt Bild 5. Bild 4 demonstriert die perspektivische Darstellung einer mathematischen Funktion, die durch Einbinden des $K8917 in eine entsprechende FORTRAN-Prozedur /4/ entstand. Bild 6 enthält ein Beispiel für den Aufruf des CZPLOT von der Betriebssystem-Ebene aus.

Installation

Für die Nachnutzung werden die Objekt-Codes der Programmteile, die dazugehörende Dokumentation, eine lauffähige gebundene Prozedur und ein Demonstrations-Bildfile auf nutzereigenen Disketten 8 oder 5,25 Zoll im Format CZ-SDOS oder UDOS bereitgestellt. Der Handler $K8917 benötigt zur Ausgabe an das RSG eine IFSS-Schnittstelle, die durch die nutzereigene Hardware zu realisieren ist. Die Hardcopy-Ausgabe vom Bildwiederholspeicher des RSG wurde mit dem Drucker robotron SD 1157/269 erprobt, vgl. Bilder 2, 3 und 4.
Das Programmsystem CZPLOT belegt in der zur Nachnutzung angebotenen Version den RAM von 5000H bis EAFFH. Der Handler $K8917 wird auch für die RAM-Belegung D000H ... EAFFH speziell für Solobetrieb bereitgestellt, damit beim Einbinden z. B. in FORTRAN oder BASIC genügend Speicherplatz für Anwender-Programme bleibt.

Grenzen und Ausblick

Natürlich sind die Möglichkeiten eines 8-Bit-Systems mit 64 KByte adressierbarem Speicher bei der Grafik-Arbeit begrenzt. Aus diesem Grund konnten im System CZPLOT folgende Funktionen *nicht implementiert* werden:

- Die Arbeit mit Segmenten (Löschen, Translation, Rotation und Skalierung)
- Funktionen, die auch das RSG nicht unterstützt, wie Schraffieren, Füllen und Routinen für Hidden-Line-Techniken usw.

Die Erprobung des CZPLOT führte aber auch zu der Erkenntnis, daß die Möglichkeiten des RSG K 8917 für interaktive Bildschirmarbeit (Grafik-Editor) durch ein 8-Bit-System in vielen Fällen sinnvoll genutzt werden können. Von der Technischen Universität Dresden wird für solche Vorhaben eine GKS-Implementierung zur Nachnutzung auf dem A 5120 angeboten (GKS = Grafisches Kernsystem /5/). Mit einem dafür geschriebenen FORTRAN-Programm konnte inzwischen ein Grafik-Editor realisiert werden, dessen Leistungsumfang in einem weiteren Beitrag vorgestellt werden soll und der vom VEB Carl Zeiss JENA ebenfalls zur Nachnutzung angeboten wird.

Literatur

/1/ Rechnergestütztes Stereokartiersystem. Druckschriften-Nr.: 14 – 345 – 1, Kombinat VEB Carl Zeiss JENA
/2/ Betriebsdokumentation K 8917. VEB Robotron-Elektronik
/3/ Gebrauchsanleitung DZT 90 × 120/RS. Druckschriften-Nr.: 14 – G 345/1 – 1, Kombinat VEB Carl Zeiss JENA
/4/ Bechstein, J.: Perspektivische Darstellung von Funktionen zweier Veränderlicher mit Hilfe von Plotter und Display (ZfK – 335). Zentralinstitut für Kernforschung Rossendorf. Juli 1977
/5/ Encarnacao, J.; Straßer, W.: Geräteunabhängige Graphische Systeme (Drittes Darmstädter Kolloquium). R. Oldenbourg Verlag München Wien 1981

⊠ KONTAKT ☏

Kombinat VEB Carl Zeiss JENA, Forschungszentrum, Carl-Zeiss-Str. 1, Jena, 6900; Tel. 833156

Termine

33. Internationales Wissenschaftliches Kolloquium
WER? Technische Hochschule Ilmenau
WANN? 24. bis 28. Oktober 1988
WO? Ilmenau
WAS?
Problemkreise der
– technischen und biomedizinischen Kybernetik
– Mathematik, Rechentechnik und ökonomischen Kybernetik
WIE? Weitere Informationen über:
Technische Hochschule Ilmenau, Direktorat für internationale Beziehungen, Vorbereitungskomitee 33. IWK, PSF 327, Ilmenau, 6300

Dr. Friedrich

10. Wissenschaftlich-technische Konferenz Elektronische Automatisierungssysteme
WER? Fachausschuß Automatisierungssysteme der Wissenschaftlich-Technischen Gesellschaft für Meß- und Automatisierungstechnik der Kammer der Technik
WANN? 28. bis 29. Oktober 1987
WO? Warnemünde
WAS?
- Neue Automatisierungsgeräte und -systeme
- Expertensysteme in der Automatisierungstechnik
- elektronische Schiffsautomatisierung als Anwendungsschwerpunkt

WIE? Teilnahmemeldungen schriftlich an: Präsidium der KDT, Sekretariat der WGMA, PSF 1315, Berlin, 1086

Müller

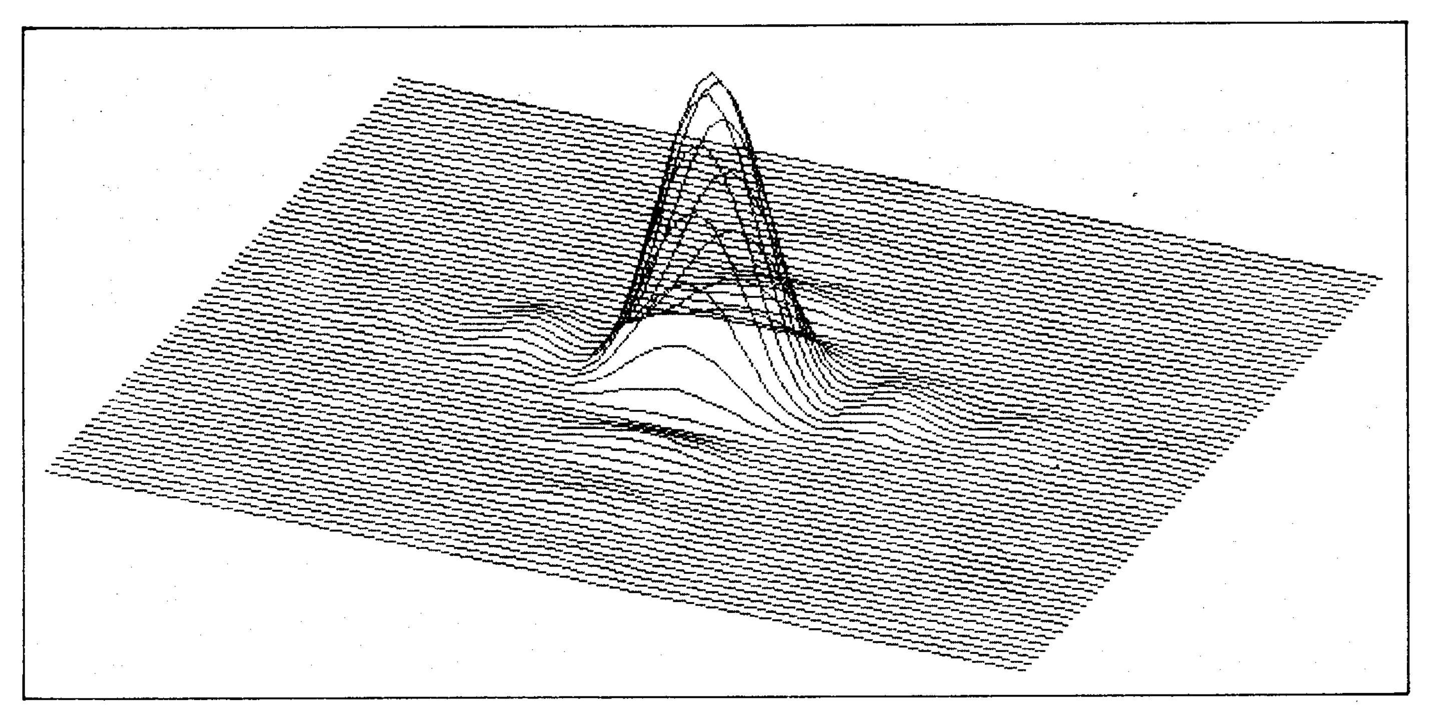


Bild 2 Drahtmodell einer mathematischen Funktion

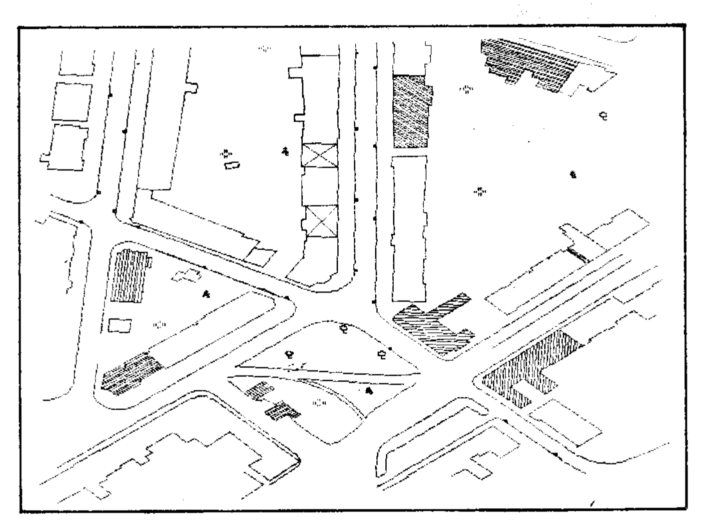


Bild 3 Arbeitsphase für einen Lageplan

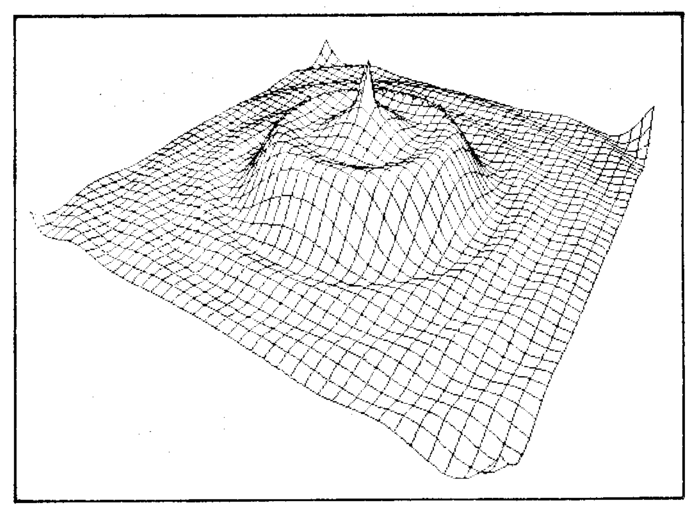


Bild 4 Perspektivdarstellung einer Funktion

```
  90 REM  HANDLER ALS PSEUDO=FILE:
 100 FILE #2;"$K8917"
 105 REM INITIALISIEREN:
 110 PRINT #2;"RS 0"
 120 PRINT #2;"WD 1 1"
 130 PRINT #2;"WW 0 1 0 1"
 140 PRINT #2;"WV 0 0.210 0 0.210"
 160 REM  DIE AUSGABESCHLEIFEN:
 170 FOR Y=15 TO -15 STEP -0.5
 180  LET X=-15
 190  GOSUB 1000
 200  PRINT #2;"P ";P;" ";Q
 210  FOR X=-14.5 TO 15 STEP 0.5
 220   GOSUB 1000
 230   PRINT #2;"S "; USING "###.##",P;" "; USING "###.##",Q
 240  NEXT X
 250 NEXT Y
 260 CLOSE
 270 END
 999 REM  FUNKTION UND ANPASSUNG:
1000 LET H=1
1010 IF X <>.O THEN LET H=SIN(.8*X)/(.8*X)
1020 LET K=1
1030 IF Y <> O THEN LET K=SIN(.8*Y)/(.8*Y)
1040 LET Z=22*H*H*K*K
1045 REM PLOTKOORDINATEN AUS DEN FUNKTIONSWERTEN:
1050 LET P=(X+15)*5+(Y+15)*1.8
1060 LET Q=55+(Y+15)*2+2.1*Z-X
1070 RETURN
```

Hinweis:
Wegen der USING-Anweisung in Zeile 230 werden nur 2 Nachkomma-Stellen ausgegeben und damit bei File-Arbeit Platz gespart.

Aufruf:

```
%CZPLOT S=1 D=$K8917 TESTBILD
```

Wirkung:

CZPLOT wird geladen und die Kommandozeile geprüft und interpretiert. Danach werden der Handler $K8917 nachgeladen, auf der Diskette in Drive 1 das File TESTBILD gesucht und der Handeingabe-Mode aufgerufen.
Wird nach Ende der Einzelbefehl-Eingabe (z.B. zum Setzen von Anfangswerten für Maßstab und Nullpunktlage) die Filearbeit gestartet, dann werden die Grafik-Befehle im ASCII-Format aus dem File TESTBILD in einen Pufferspeicher eingelesen. Jeder Befehl wird interpretierend bearbeitet und, falls er zur Menge des in Tafel 1 gezeigten Befehlssatzes gehört, in entsprechend bearbeiteter Form dem Handler $K8917 zur Ausgabe übermittelt.
Nach Ende der Filearbeit kann wieder zum Handeingabe-Mode gegangen und z.B. ein neuer Maßstab oder Nullpunkt festgelegt und das File erneut gelesen werden.

S=SOURCE (Quelle) ist hier Laufwerk (DRIVE) Nr.1
D=DESTINATION (Bestimmung,Ziel) ist hier Handler $K8917,
  Kann sonst jedoch auch ein Laufwerk sein, auf dessen
  Diskette dann ein neues File angelegt wird, in dem die
  Ausgabe-Befehle eingetragen werden.

Bild 5 Beispiel für die Einbindung des $BA8917 in BASIC
(Der Funktionsverlauf ist im Bild 2 dargestellt.)
Bild 6 Beispiel für Aufruf des CZPLOT

ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 7
S. 193–224

Auslieferung in diesen Tagen durch den Fachbuchhandel



Taschenbuch Elektrotechnik in sechs Bänden

Band 2:
Grundlagen der Informationstechnik

Herausgegeben von Prof. em. Dr. sc. techn. Dr. techn. h. c. Eugen Philippow. 3., stark bearbeitete Auflage. 984 Seiten, 836 Bilder, 161 Tafeln, Kunstleder, DDR 40,– M, Ausland 53,– DM. Bestellangaben: 553 615 9/ Tb. Elektro 2

Die Auflage wurde völlig überarbeitet und auf den neuesten Stand gebracht. Darüber hinaus wurde sie um einige Abschnitte erweitert, die wichtige neue Berechnungsverfahren und in letzter Zeit besonders aktuell gewordene Gebiete zum Inhalt haben.

Inhaltsübersicht: Allgemeine Systemtheorie · Angewandte Systemtheorie – Signale und lineare Systeme · Informationstheorie · Kodierungs- und Modulationstheorie · Kontinuierliche Steuerungssysteme · Diskontinuierliche Steuerungssysteme · Experimentelle Prozeßanalyse · Frequenzselektive Netzwerke · Nichtlineare und parametrische Netzwerke · Digitale Schaltungen · Elektromagnetische Wellen · Elektroakustik · Bedienungstheorie · Zuverlässigkeitstheorie · Empfindlichkeit dynamischer Systeme · Informationsaufnahme, -verarbeitung und -speicherung sowie Steuerungsprozesse in Lebewesen.

Elektronische Schaltungstechnik

Von Prof. Dr. sc. techn. Roland Köster und Prof. Dr.-Ing. habil. Albrecht Möschwitzer. 4., durchgesehene Auflage. 304 Seiten, 344 Bilder, 32 Tafeln, Leinen, DDR 42,– M, Ausland 42,– DM. Bestellangaben: 553 458 3/ Köstner, Schaltung

Dieses Buch ist eine Einführung in das umfangreiche Gebiet der analogen und digitalen Schaltungstechnik, und zwar in erster Linie in Hinblick auf die moderne technologische Realisierung in Form von integrierten Schaltkreisen.

Einführend wird eine kurze Beschreibung der Eigenschaften der wichtigsten Bauelemente, der Stromkreisberechnung und des Rauschverhaltens gegeben. Daran schließen sich die getrennten Abschnitte zur Analogtechnik und zur Digitaltechnik an. Es erfolgt eine ausführliche Behandlung der Grundschaltungen, wobei auf heute wesentliche Schaltungsprinzipien orientiert wird. Auf diesen Grundlagen aufbauend, werden die am häufigsten verwendeten Schaltungskomplexe, so wie sie als integrierte Schaltkreise angeboten werden, in ihrer Funktion beschrieben und Fragen der optimalen äußeren Beschaltung behandelt. Dabei wird im digitalen Teil auch auf hochintegrierte Rechnerschaltkreise (Mikroprozessoren, Speicher) eingegangen. Bei den Schaltkreisen beziehen sich die Autoren auf Applikationshinweise der Hersteller und auf eigene Erprobungsergebnisse.

TECHNIK-WÖRTERBUCH
Polytechnisches Wörterbuch

Englisch-Deutsch. Herausgegeben von Ing. Rudolf Walther. 6., unveränderte Auflage. 1128 Seiten, Kunstleder, DDR 62,– M, Ausland 80,– DM. Bestellangaben: 553 358 0/Walther, Polyt. Wb. E-D
Mit etwa 100 000 Wortstellen sowie 3 000 Abkürzungen.

TECHNIK-WÖRTERBUCH
Hochenergiephysik

Englisch-Deutsch-Französisch-Russisch. Zusammengestellt von Dipl.-Math. Ralf Sube. 1. Auflage. 164 Seiten, Kunstleder, DDR 22,– M, Ausland 36,– DM. Bestellangaben: 553 764 5/Sube, Hochenergiephysik

Mit etwa 4 500 Wortstellen u. a. aus den Gebieten Elementarteilchen, Feldtheorien, Nachweis und Messung von Elementarteilchen, Teilchenbeschleuniger und deren Untergebieten.
Das Wortgut entspricht dem derzeitig wissenschaftlichen Stand des Fachgebietes und zeichnet sich durch hohe fachspezifische Genauigkeit aus.

Zusammen mit den vom gleichen Autor erschienenen Bänden „Kerntechnik" (Bestellnummer 553 357 2) und „Strahlenschutz, Strahlenbiologie, Nuklearmedizin" (Bestellnummer 553 479 4) ist nun das Gesamtgebiet der Kernphysik und ihrer Anwendung erfaßt.

VEB VERLAG TECHNIK BERLIN



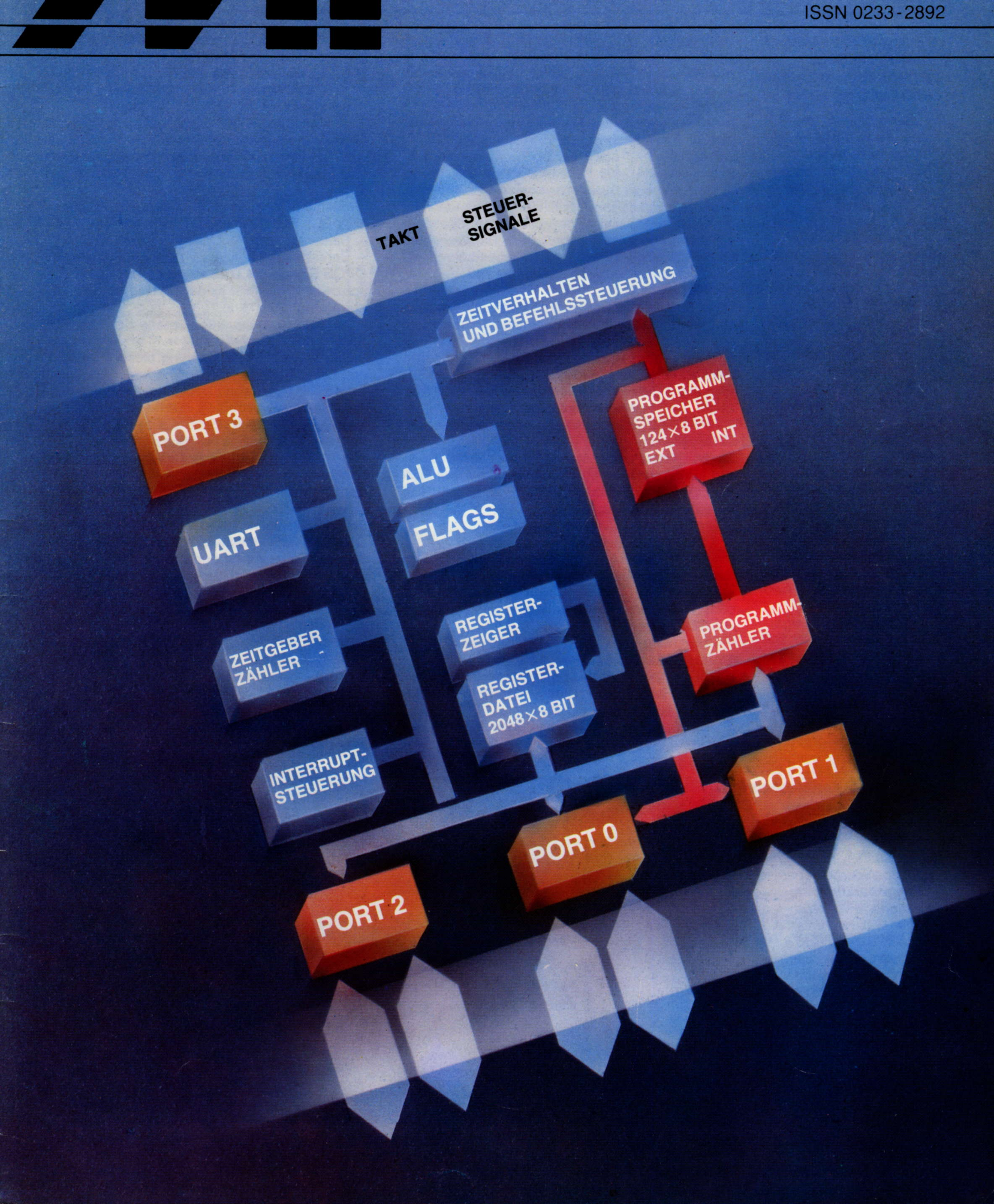
ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

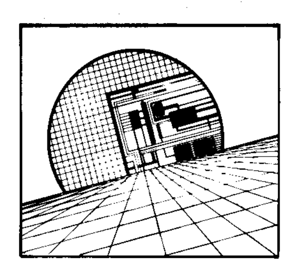
MP

Heft 8 · 1987

Mikroprozessortechnik

VEB Verlag Technik Berlin
ISSN 0233-2892





12. Mikroelektronik-Bauelemente-Symposium

12. Mikroelektronik-Bauelemente-Symposium in Zahlen und Fakten

– Gemeinsame Veranstaltung des VEB Kombinat Mikroelektronik und des Bezirksvorstandes Frankfurt (Oder) der Kammer der Technik
– Seit 1966 mit 300 Teilnehmern als Halbleiterbauelemente-Symposium in Frankfurt (Oder) durchgeführt
– Ab 1985 unter der Schirmherrschaft des Ministers für Elektrotechnik/Elektronik als Mikroelektronik-Bauelemente-Symposium
– 12. MEBS und Fachtagung 1987 mit 2400 Teilnehmern, darunter über 2000 Vertreter aus bauelementeherstellenden und Hauptanwenderbetrieben, davon 500 aus Betrieben des Maschinenbaus, über 100 Teilnehmer von Einrichtungen der AdW der DDR, 250 Gäste aus Instituten, Universitäten, Hoch- und Fachschulen
– Die Ausstellung zeigte auf 1200 m² Fläche das verfügbare Bauelementesortiment und die breite volkswirtschaftliche Anwendung am Beispiel von 160 Exponaten
– 16 Fachvorträge wurden zum Symposium gehalten, darunter 28 Vorträge von Vertretern bauelementeherstellender Betriebe, 13 Vorträge von Referenten aus Anwenderbetrieben, Universitäten, Instituten und Ingenieurbetrieben für angewandte Mikroelektronik, 4 Vorträge hielten Gastreferenten aus der UdSSR und ČSSR
– Umfangreicher Erfahrungsaustausch fand in Posterdikussionen im Rahmen der Applikationsausstellung und an Konsultationsstützpunkten von Bauelementeherstellern statt.

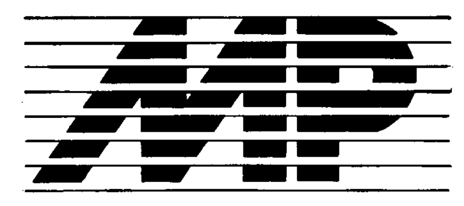
Mehr als 2400 Wissenschaftler und Praktiker aus allen Bereichen der Volkswirtschaft nahmen am Symposium – einer der größten Informationsveranstaltungen zu Entwicklung, Produktion und Anwendung der Mikroelektronik in der DDR – teil.
In seinem einleitenden Referat unterstrich der Minister für Elektrotechnik und Elektronik, Felix Meier, daß sich bis 1990 das in der DDR vorhandene Sortiment von rund 1400 Bauelementetypen durch die Einführung neuer Basistechnologien erweitern wird. Die wissenschaftliche Arbeit muß insbesondere auf die Entwicklung von Mikroprozessoren, Speicherschaltkreisen, optoelektronischen Bauelementen für die Lichteitertechnik, Display-Farbbildröhren und oberflächenmontierbaren Bauelementen gerichtet werden.
In der parallel zum Symposium stattfindenden Ausstellung (Bild 1) zeigten das Kombinat Mikroelektronik, Anwenderbetriebe und wissenschaftliche Einrichtungen etwa 160 Exponate. Einige davon sollen hier kurz vorgestellt werden.
Eine *256-KByte-DRAM-Baugruppe für den K1520* zeigte die Ingenieurschule für Seefahrt Warnemünde (Bild 2). Nähere Angaben dazu sind dem Bild zu entnehmen.
Der VEB Forschungszentrum Mikroelektronik Dresden (ZMD) stellte einen *16-KByte-Arbeitsspeichermodul für K1520-Anwendungen* vor (Bild 3). Die Kurzcharakteristik lautet:
– Statischer 16-KByte-Arbeitsspeicher
– K1520-Schnittstelle
– freie Wahl der Anfangsadresse in 4-KByte-Segmenten innerhalb des U880-Adreßraumes
– Datenschutz gegen Nennspannungsausfall durch Betriebsspannungspufferung mit 3 Ni-Cd-Akkus GLZ 225 mAh
– Gewinnung eines WAIT-Zyklus ist möglich.
Als Bauelementebasis werden benötigt:
U214D, U224D, U2148D (je nach Erfordernis); DS-Interface-Reihe; bei Pufferung der Betriebsspannung V4066D, B340D, B081D.
Ebenfalls große volkswirtschaftliche Effekte verspricht die Nutzung des *Floppy-Disk-Controller-Moduls K5126* des VEB Robotron Buchungsmaschinenwerk Karl-Marx-Stadt (Bild 4).
Die Charakteristik des *IEC-Bus-Interface U5201 PC-103* (Bild 5) des VEB ZMD läßt sich wie folgt kurz zusammenfassen:
– kundenspezifischer Schaltkreis auf Basis des Gate-Array-Systems U5200, konzipiert für die Realisierungen eines L/T-Interface nach TGL 42039 bzw. IEC-Standard 625 in Kleinserien-Labormeßgeräten.
– Interface-Funktionen: SH1, AH1, T1, T5, L1, L3, SR1-RL1, PP1, DC1 und DT1
– Steuereingänge für Controllereinsatz
– Bereitstellung eines handshake-gesteuerten OPQS-Signals mit Treibern, so daß im Trivialfall ein spezieller Multiplexer für das Statusbyte der Serienabfrage entfallen kann
– handshake-gesteuerte Ausgabe der End-Nachricht.
Bisher waren zur Realisierung dieser Funktionen drei Leiterkarten (siehe Bild 5) erforderlich. Weiterhin im Bild 5 zu sehen (rechts) der kundenspezifische Steuerschaltkreis PL 5201-112 auf Basis des Gate-Array-Systems U5200 aus dem Entwurfszentrum des VEB Textima Karl-Marx-Stadt.
Bild 6 zeigt den Arbeitsplatzrechner *Scheibeninspektion*.
Der Computer bewertet 100-mm-Scheiben mit gut/schlecht und übernimmt die statistische Auswertung der angefallenen Meßdaten. Entwickler dieses Computers zur Rationalisierung der Fertigung von 100-mm-Scheiben ist der Ingenieurbetrieb Mikroelektronik Frankfurt (Oder).
In einem Ausstellungskomplex wurde gesondert auf die überaus wichtige Thematik der kundenspezifischen Schaltkreise eingegangen. Die beiden nachfolgenden Übersichten entnahmen wir diesem Ausstellungskomplex.
Aus der Übersicht 1 sind die Leistungen des VEB ZMD und die Anforderungen an die Anwender ersichtlich. Die Übersicht 2 liefert eine Gegenüberstellung des Gate-Array-Systems mit dem Standardzellen-System.

I. P.

Fortsetzung auf der 3. Umschlagseite

1


Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0233–2892



Mikroprozessortechnik, Heft 8 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 2 87 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)
Titel: Tatjana Stephanowitz

Beirat Dr. Ludwig Claßen, Prof. Dr. sc. Dietrich Eckhardt, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Jürgen Zaremba

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

Redaktionsschluß: 22. Juni 1987

AN (EDV) 49837

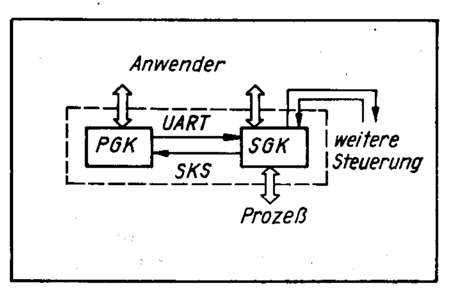
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010, und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig

Adresse	Programmspeicher	Datenspeicher
%FFFF / %E000	u 2365 - BM 200 (ROM)	
%DFFF / %6000	Externe Speichererweiterung	
%5FFF / %4000	Adreßraum für Bausteine (MEMORY-MAPPED I/O)	
%3FFF / %2800	Arbeitsspeicher (extern)	
%27FF / %2000	Arbeitsspeicher auf EM vorhanden	
%1FFF / %0800	RAM für Programmentwicklung	EPROM für entwickelte Programme
%07FF / %0000	interner u 883ROM	nicht nutzbar

Seite 232



Seite 235

Struktogramm | Umsetzung in REDABAS
(anweisung_1)
(anweisung_2)
:
(anweisung_n)

Seite 239

Vorschau

Für MP 9/87 haben wir für Sie u. a. Beiträge zu folgenden Themen vorgesehen:
– Analogwerterfassung mit den A/D-Wandlern C571/570
– Zeitoptimierte A/D-D/A-Baugruppe mit C571/C565
– C571D an U880-Systemen
– BASIC-Sprachübersicht für KC85, KC87 und SCP-BASIC-Interpreter

Inhalt



Info

Patentanmeldungen durch gemeinsame Forschungsarbeit

Koordinierungs- und Leistungsverträge verbinden das Büromaschinenwerk „Ernst Thälmann" Sömmerda mit mehr als 30 Forschungseinrichtungen und Kombinaten der DDR. An gemeinsamen Vorhaben, die zum Teil über die 90er Jahre hinausreichen, sind die Technischen Hochschulen Ilmenau und Leipzig, mehrere Universitäten sowie Institute der Akademie der Wissenschaften beteiligt. Durch langfristig angelegte Wissenschaftskooperation erweitert der Robotron-Betrieb seine Möglichkeiten zur Entwicklung neuer Erzeugnisse und effektiver Verfahren. Sie trägt dazu bei, daß jährlich mehr als 40 Prozent der Produktion den größten Industriebetrieb im Bezirk Erfurt als Neu- und Weiterentwicklung verlassen. Die Hauptrichtung gemeinsamer Forschung, die vor allem auf wissenschaftlich-technischen Vorlauf orientiert, ist die Arbeit an neuen Wirkprinzipien für leistungsfähige und variabel einsetzbare Drucktechnik. Weitere Themen beschäftigen sich mit neuen Werkstoffen sowie Softwarelösungen für den im Betrieb in steigenden Stückzahlen gefertigten PC 1715. Beispiel erfolgreicher Kooperation mit der Friedrich-Schiller-Universität Jena und der Technischen Universität Magdeburg seien Ergebnisse bei der Entwicklung nichtmechanischer Drucker auf Laserbasis. Bisher konnten von den Mitgliedern des interdisziplinären Jugendforscherkollektivs, das diese Aufgabe aus dem Plan Wissenschaft und Technik bearbeitet, 17 Erfindungen zum Patent angemeldet werden. Bewährt hat sich auch der Erfahrungsaustausch bei der Einführung von 331 CAD/CAM-Lösungen im vergangenen Jahr. Wissenschaftler wirkten beim Aufbau der rechnergestützten Produktion von Schrittmotoren in einem automatisierten Fertigungsabschnitt mit.

ADN

Erste Softwarebörse in Berlin veranstaltet

Die erwartete große Resonanz fand am 2. Juni 1987 die 1. Softwarebörse des Stadtbezirks Berlin-Lichtenberg. Sie war von der Lichtenberger CAD/CAM-Arbeitsgemeinschaft und der Ingenieurgruppe Territoriale Rationalisierung des Rates des Stadtbezirks organisiert worden, um den Erfahrungsaustausch zur Entwicklung von Anwendungssoftware zu fördern und vor allem Klein- und Mittelbetrieben die Nachnutzung vorhandener Lösungen zu erleichtern.
Mit Unterstützung des Bezirksneuererzentrums (BNZ), in dessen Räumen die Börse auch stattfand, des Leitzentrums für Anwendungsforschung (LfA) und anderer konnten 40 Exponate aus 9 Lichtenberger Betrieben und Institutionen angeboten und in Vorträgen zu einigen Projekten Kenntnisse vermittelt werden.
Mehr als 630 Besucher aus allen Teilen der Republik – über 100 aus Berliner Einrichtungen – nutzten das Angebot, aus dem zum Beispiel die Programme Stellenplanung, Personal und Krankenstandsanalyse (VEB Herrenbekleidung Fortschritt), Technische Stammdaten (VEB Lufttechnische Anlagen Berlin), Transportkostenabrechnung (VEB Maschinenbauhandel Berlin) oder Rechnerverbund mit CCSMV (VEB LfA) besonders gefragt waren.
Die Hardwarebasis für die Programme reichte vom Kleincomputer über den Personalcomputer bis zur Kopplung mit ESER-Rechnern.
Von dem überaus großen Informationsbedürfnis zeugten die etwa 200 Eintragungen zur Anforderung weiterführender Informationen, wenngleich sie zum Teil auch darin begründet waren, daß die vorbereiteten Dokumentationszettel nur kurze Zeit reichten. Diese und weitere Erfahrungen aus der Berliner Softwarebörsen-Premiere sollen bei der Vorbereitung der 2. Lichtenberger Börse berücksichtigt werden, die für Ende des Jahres – also nach der Berliner Softwarebörse der KDT – vorgesehen ist.

MP

Interrupt

Fast wie ein Interrupt für die redaktionelle Arbeit unserer Zeitschrift MP wirkt z. Z. die unbesetzte Stelle einer Redaktionssachbearbeiterin bzw. eines -sachbearbeiters.
Falls Sie Interesse an dieser Tätigkeit haben, gerne mit moderner Technik arbeiten und im Raum Berlin wohnen, rufen Sie uns unter Tel. 2870203 oder 2870371 an.
Gleichzeitig möchten wir all jene Leser, die uns geschrieben haben, dafür um Verständnis bitten, daß Sie etwas länger als gewohnt auf unsere Antwort warten müssen.

Ihre Redaktion MP

Softwarekatalog für Halle

Die Stadt Halle besitzt ihren ersten Softwarekatalog. Die Programmbibliothek speichert gegenwärtig über 150 zumeist nachnutzungsfähige Anwenderprogramme für Bürocomputer, die in Betrieben und Einrichtungen der Saalestadt schon im Einsatz sind. Es handelt sich dabei vor allem um Software zur Leitung und Planung. Damit können Abrechnungen, Materialdispositionen und Verwaltungsarbeiten rationeller gestaltet werden. Künftig werden dort auch CAD-Lösungen der Betriebe erfaßt. Um die Arbeit mit dem Katalog, über den derzeit bereits rund 40 Betriebe verfügen, zu erleichtern, kommt ein Rechercheprogramm aus dem VEB Ingenieurtechnik Halle als Nachnutzung zur Anwendung. Das ermöglicht, anhand ausgewählter Stichwörter und Schlüsselnummern rasch und zielgerichtet per Rechner Angaben über das Angebot zu erhalten. Der jährlich vervollständigte Katalog soll künftig auch Grundlage einer Softwarebörse sein.

ADN

Schaltkreise nach Kundenwunsch

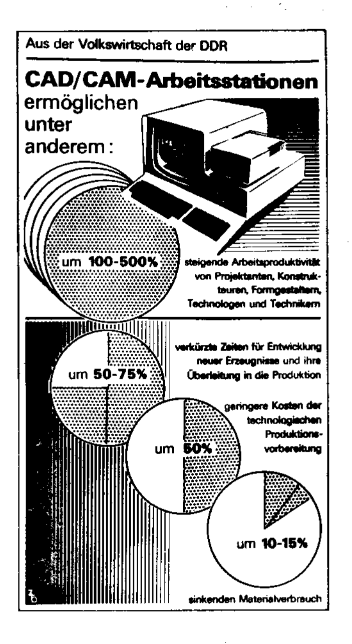
Rund 50000 Schaltkreise nach Kundenwunsch (Gate Array) werden 1987 vom Forschungszentrum Mikroelektronik in Dresden hergestellt. Der zum Kombinat Carl Zeiss JENA gehörende Betrieb erprobt gegenwärtig für deren Produktion eine neugestaltete, bisher für Forschungsaufgaben eingesetzte technologische Linie. Sie soll Mitte des Jahres die Serienfertigung dieser auf die Erfordernisse der Kunden zugeschnittenen Schaltkreise übernehmen. Erfahrungen besagen, daß sich gegenüber herkömmlichen Lösungen die Entwurfs- und Fertigungszeiten für einen Gate-Array-Schaltkreis auf etwa ein Viertel verringern lassen.

ADN

CAD/CAM-Arbeitsstationen

Bis 1990 werden in der Volkswirtschaft der DDR 85000 bis 90000 CAD/CAM-Arbeitsstationen wirksam werden, an denen insbesondere Konstrukteure, Projektanten, Technologen, Formgestalter und weitere ingenieurtechnische Kader tätig sind. Die Grafik stellt einige Effekte dar, die mit dem Einsatz erzielt werden.

Grafik: ADN/ZB



Joystick

Vom VEB Meßtechnik Mellenbach und Fachleuten der Friedrich-Schiller-Universität Jena wurde ein Joystick entwickelt. Es hat die Größe einer Zigarettenschachtel. Mit Joysticks kann man auf dem Bildschirm von Mikrorechnern oder Heimcomputern Freihandzeichnen, schreiben oder auch Symbole aus gespeicherten Bauteillisten zum Beispiel in Leiterplattenentwürfe übertragen. Grundlage des Prinzips war die Entwicklung völlig neuer Magnetfeldsensoren. Der Joystick läßt sich mit allen in der DDR produzierten Rechnertypen koppeln. Sie müssen über einen PIO-Anschluß verfügen. Die ersten 500 Stück sollen noch in diesem Jahr ausgeliefert werden.

ADN

Kabelloser Drucker

Der HP 82240A von der Firma Hewlett-Packard, der Ende vergangenen Jahres erstmals vorgestellt wurde, druckt im Thermoverfahren eine 24 Zeichen lange Zeile in knapp einer Sekunde. Das Besondere – er empfängt die auszugebenden Daten mittels Infrarotsignale. Die Stromversorgung erfolgt über vier Alkalibatterien – in diesem Fall kommt er ohne jede Kabelverbindung aus – oder über einen Netzadapter. Die Datenübertragung vom Rechner zum Drucker ist bis zu einer Entfernung von 1 Meter möglich.

USA: Aufkäufe von Computerfirmen

Mit der Übernahme des Unternehmens Uccel Corp. für 800 Mio. Dollar ist die USA-Firma Computer Associates International Inc. zum größten Software-Lieferanten auf dem kapitalistischen Weltmarkt geworden. Das international noch wenig bekannte Unternehmen entwickelt vorrangig Systemsoftware für IBM- und IBM-kompatible Rechnersysteme. 1986 erzielte es bei einem Gesamtumsatz von 309 Mio. Dollar einen Reingewinn von 36,4 Mio. Dollar. Seit 1981 übernahm die Firma insgesamt sieben kleinere Unternehmen der Computerbranche in den USA.
Der PC-Hersteller Computerland Corp. ist von der US-amerikanischen Investmentbank E. M. Warburg, Pincus & Co. aufgekauft worden. Der Kaufpreis soll zwischen 200 und 250 Mio. Dollar gelegen haben.

Computerzeitschriften in der VR Polen

KOMPUTER heißt eine populärwissenschaftliche Monatszeitschrift, die seit Anfang 1986 in unserem Nachbarland herausgegeben wird. Die Zeitschrift ist einerseits für fortgeschrittene Nutzer von Mikrocomputern gedacht – andererseits soll sie aber auch breitere Leserkreise ansprechen. Es werden neueste Geräteentwicklungen vorgestellt und deren Programmierung beschrieben. Ebenfalls auf großes Interesse, vor allem bei jugendlichen Lesern stößt die Zeitschrift Bajtek. Sie bietet besonders für den an Heimcomputertechnik Interessierten Gerätevorstellung und Programmveröffentlichungen.

Repro: Wosnizok



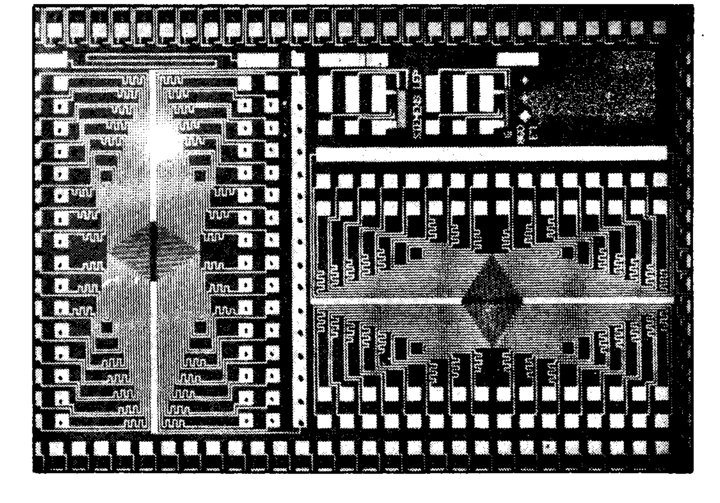
Grafikcode

Die japanische Firma Jatech, Produzent von Verpackungsmaterial, will Grafik-Symbole als Code für die Kennzeichnung von Waren verwenden. Grafikcode bietet gegenüber den bisher häufig genutzten Strichcodes (Barcode) einige Vorteile. Jedes Grafik-Symbol kann 16 verschiedene Informationen aufnehmen (siehe auch Bild 5 auf Seite 250) – beim Barcode sind es nur zehn. Außerdem dürfen die Striche des Barcodes nur um 0,01 mm tolerieren, während die Grafik-Symbole bis zu 0,75 mm außerhalb der Soll-Position liegen können. Wegen dieser geringeren Anforderungen sollen die Kosten beim Druck des Grafikcodes bedeutend niedriger sein als beim Barcode. Größter Vorteil des neuen Codes dürfte sein, daß die Felder auch von Hand ausgefüllt werden können. *MP*

Integrierte Schaltungen auf GaAs

GaAs wird als Halbleitermaterial immer wichtiger. Auf Grund der höheren Grenzfrequenz und der günstigeren Rauscheigenschaften im Vergleich zu Silizium finden Transistoren aus GaAs bei digitalen und analogen Schaltungen in Datenverarbeitung und Nachrichtentechnik mehr und mehr Verbreitung.
Verwendet werden bisher weitgehend Einzeltransistoren oder niedrig integrierte Bausteine. Die maximal erreichte Integrationsdichte von digitalen GaAs-Chips liegt in der Fertigung bei 1000 Gattern, im Gegensatz zu Silizium, wo heute 2 Millionen Transistoren pro Chip möglich sind und 8 Millionen Transistoren pro Chip im Bereich des technologisch Möglichen liegen. Der Grund hierfür liegt neben der schwierigen Prozeßtechnologie auch an der Qualität des Ausgangsmaterials selbst. Anders als Silizium läßt sich GaAs nicht so leicht in der notwendigen Reinheit, Defektfreiheit und Homogenität der elektrischen Eigenschaften herstellen.
Fertigt man auf einer GaAs-Scheibe eine große Anzahl komplexer integrierter Schaltungen, war man nicht sicher, daß deren Eigenschaften an verschiedenen Stellen der Scheibe identisch sind. Bei Schaltungen mit engen Fertigungstoleranzen stieg der Ausschuß stark an. Es gab bisher keine fertigungsgerechte Möglichkeit, einer GaAs-Scheibe „im vorhinein" anzusetzen, ob sie eine homogene Defekt- und Verunreinigungsdichte aufweist oder nicht. Ein in den Siemens-Laboratorien zusammen mit Philips (Frankreich) entwickeltes Verfahren schaffte hier Abhilfe: Ausgangspunkt der Überlegungen war der experimentelle Befund, daß eine oder maximal zwei Scheiben für einen ganzen GaAs-Stab repräsentativ sind. Durch eingehende Untersuchungen konnte ferner gezeigt werden, daß die lokale Defektdichte mit der Einsatzspannung von MESFET-Transistoren korreliert ist.
Zur Charakterisierung eines GaAs-Stabs wird auf eine Scheibe des Stabes eine Anzahl von Teststrukturen aufgebracht, bei denen jeweils auf eine Länge von 300 μm 30 Transistoren im Abstand von nur 5 μm nebeneinander angeordnet sind, d. h. in der Teststruktur werden IC-typische Dimensionen erreicht. Die großen Anschlußkontakte ordnet man relativ weit vom eigentlichen Transistor an und winkelt die Zuführungen so, daß alle Transistoren den gleichen effektiven Kontaktwiderstand haben. Mit einer solchen Struktur ist es möglich, die Verteilung der Einsatzspannungen der Testtransistoren automatisch mit einem rechnergesteuerten Tester zu messen und so die Auswirkung der Verunreinigungen und Defekte auf Transistoreigenschaften in einem Raster von unter 10 μ zu prüfen.

Werkfoto



MP-Computerclub

Auf vielfältigen Wunsch beginnen wir in MP 9/87 mit unserer neuen ständigen Rubrik MP-Computerclub, in der wir Hard- und Softwarelösungen zu Kleincomputern vorstellen werden. Gefragt sind dazu Tips und Tricks für den Umgang mit dem Computer. Gern berichten wir auch über Erfahrungen der Computerclubs und -kabinette.
Senden Sie uns also unter dem Kennwort „MP-Computerclub" Ihre Tips und Anregungen.
Unsere Anschrift lautet:
VEB Verlag Technik, Redaktion MP, Oranienburger Str. 13/14, Berlin, 1020

FORTH-83

Das im Beitrag „FORTH: Eine außergewöhnliche Softwarekonzeption" von G.-U. Vack in MP 6/87 angekündigte Material kann über folgende Adresse bestellt werden:
KDT Suhl, Straße der OdF 29, Suhl, 6000.
Die Broschüre enthält die Original-Spezifikation des Standards FORTH-83 sowie einige Empfehlungen zur Programmgestaltung.



Dialog

Liebe Leser,

stellvertretend für die vielen Zuschriften, die die Redaktion erreichten, veröffentlichten wir in MP 6/87 eine kleine Auswahl. Sie spiegelt die Breite der Meinungen allerdings nur unvollständig wider.
Themenwahl und fachliches Niveau standen in den Zuschriften im Mittelpunkt des Interesses. Einige wünschen sich eine reine Hardware-(Schaltkreis-)Zeitschrift, andere wollen in erster Linie Softwareaspekte berücksichtigt sehen.
Viele beklagten sich über das „hohe wissenschaftliche Niveau" der ersten Ausgaben und waren der Meinung, eine Zeitschrift mit populärwissenschaftlicher Orientierung sei das Richtige. Häufig wurden in den Briefen an die Redaktion auch Fragen zum Veröffentlichungsprofil gestellt.
Kritikwürdig war – nach Ansicht vieler Leser – die großzügige Gestaltung von Heft 1 und 2. Lob und Kritik gleichzeitig gab es für die Autorenvorstellung in Wort und Bild. Die einen meinten, weitere Fachzeitschriften könnten sich dieser Form der Autorenvorstellung anschließen, andere Leserstimmen bezeichneten es als „Platzverschwendung".
Apropos Platz – der aufmerksame Leser wird festgestellt haben, daß ab MP 6/87 auch im Innenteil der gleiche Schriftgrad – also ein kleinerer als bisher – wie im Rubrikenteil verwendet wird. Das bedeutet: ab Heft 6 mehr Informationen in jeder Ausgabe. Aus dem gleichen Grund wurde ab Heft 3 auf die Bildvorstellung der Autoren verzichtet.
Vollständig die Themen anzuführen, denen MP sich zuwenden wird, würde an dieser Stelle den Rahmen sprengen. Die Zielstellung unserer Zeitschrift hat der Minister für Elektrotechnik und Elektronik der DDR, Felix Meier, in seinem Leitartikel in MP 1/87 dargelegt. In MP 2/87 wurden im redaktionellen Vorwort die Schwerpunktthemen kurz umrissen und das Anliegen der Rubriken erläutert.

Einheit von Hard- und Software

Dennoch soll hier aufgrund der zahlreichen Anfragen kurz auf einige Aufgaben der Zeitschrift MP eingegangen werden. Die bedeutendste Applikation der Mikroelektronik ist die Mikroprozessortechnik. Deshalb nehmen Veröffentlichungen über Mikroprozessorsysteme – Mikroprozessoren und die dazugehörigen Bauelemente, Baugruppen bzw. Komponenten – einen wichtigen Platz in der Zeitschrift ein. Dabei werden Hardwarevorstellungen, Applikationsmöglichkeiten und Fragen der Programmierung gleichermaßen berücksichtigt. Hierbei in MP nur die Softwaregesichtspunkte zu betrachten und zu Hardwareproblemen auf andere Zeitschriften zu verweisen, ist wenig sinnvoll und würde die fachliche Aussage beeinträchtigen.
Einen weiteren Schwerpunkt bilden die Veröffentlichungen zu Mikrocomputern. Dabei ist es gleichgültig, ob es sich um AC, BC, PC oder KC handelt. Hardwareerweiterungen sind genauso gefragt wie Software-Tips und Tricks.

Veröffentlichung von Programmen

Gern drucken wir kurze Programme ab. Voraussetzung dafür ist, daß es sich um solche Lösungen handelt, die für einen großen Leserkreis von Interesse sind. Vorteile sind die sofortige Nachnutzbarkeit und methodische Effekte. Wir sehen es als Selbstverständlichkeit an, daß der Autor sein Programm umfassend getestet hat und Rechte Dritter an der angebotenen Lösung nicht bestehen bzw. falls die Software im Rahmen einer Dienstaufgabe erstellt wurde, die Freigabe durch die Dienststelle erfolgt ist und beiliegt. Was wir natürlich nicht möchten, von irgendwoher „gekupferte" Programme veröffentlichen. Das bringt unser Anliegen in Mißkredit. Wir bitten daher, davon abzusehen, der Redaktion derartige „Schöpfungen" einzureichen. Umfangreiche Programme lassen sich aus Platzgründen nicht abdrucken. Betriebe können ihre Hard- und Software-Lösungen sowie natürlich auch entsprechende Suchmeldungen in der Rubrik Börse offerieren. Um die Mehrfachnutzung von Software zu fördern – auch unter Berücksichtigung der hohen Auflage der MP –, ist in Börse das Bieten und Suchen von Anwendersoftware möglich, vorrangig von Lösungen zur Verbesserung der Standardsoftware (z. B. Dienst- oder Hilfsprogramme). Umfassendere Informationen zu spezieller Anwendersoftware gehören in die Zeitschriften des entsprechenden Fachgebietes.

Zeitschrift für einen breiten Kreis von Fachleuten

Grundsätzlich soll sich MP zu einer Zeitschrift für einen breiten Kreis von Fachleuten profilieren, um die umfassende Einführung der Mikroelektronik in alle Bereiche der Volkswirtschaft wirksam zu unterstützen. Das bedeutet nicht ein Herangehen an die Themen in populärwissenschaftlicher Darstellungsweise, aber es bedeutet auch nicht das Veröffentlichen von Beiträgen, die nur für wenige Spezialisten von Interesse sind.
Unsere Autoren, der Redaktionsbeirat und die Redaktion stehen vor der Aufgabe, einen entsprechenden „Mittelweg" zu finden. Daß dabei der richtige Weg beschritten wird, zeigt die ständig steigende Nachfrage nach *Mikroprozessortechnik*.

Neue Rubriken

Unter der Rubrik „Wie funktioniert ein ...?" bzw. „Was ist ein ...?" – erstmals in MP 7/87 – werden in loser Folge Begriffe leicht verständlich erklärt. Mit dieser Rubrik entsprechen wir einem vielfach geäußerten Leserwunsch. Ebenfalls auf Leserwunsch entstanden ist die in Heft 9 beginnende ständige Rubrik MP-Computerclub. Auf monatlich etwa zwei Druckseiten soll sie als Podium für Computerclubs und Computerkabinette dienen. Dabei geht es vorrangig um die Computertechnik, die in den Einrichtungen in der Regel vorhanden ist, also Kleincomputer der Kombinate Mikroelektronik und Robotron, den Mikrorechnerbausatz Z1013 und natürlich auch den PC 1715.
Außerdem in Vorbereitung befindet sich eine Rubrik, in der in kurzer Form über internationale Entwicklungstrends von Hard- und Software informiert wird.
Abschließend noch einige Worte zu Ihren Zuschriften. Jeder Brief wird in der Redaktion ausgewertet. Aber bitte haben Sie etwas Geduld, wenn wir wegen der Fülle der Einsendungen nicht in der Lage sind, auf jede Zuschrift sofort zu antworten. Wir nehmen natürlich weiterhin gern Ihre Hinweise, Ideen, Anregungen und Ihre Manuskripte entgegen. Denn um den Interessen vieler Leser gerecht werden zu können, brauchen wir auch die Mitarbeit vieler Leser. Falls Sie die Möglichkeit sehen, uns dabei zu unterstützen, die MP noch interessanter und praxiswirksamer zu gestalten, so schreiben Sie uns – oder rufen Sie einfach an!
Ihre Redaktion MP

Mitarbeit gefragt

Für eine nebenberufliche Gutachtertätigkeit zum Bewerten von Hard- und Softwarelösungen für die MP suchen wir Spezialisten – möglichst aus dem Raum Berlin, aber nicht Bedingung – zu folgenden Gebieten: PC 1715- und A 71xx-Hardware, CP/M, UNIX, MS-DOS und kompatible Betriebssysteme sowie REDABAS/dBASE II.

Bitte wenden Sie sich schriftlich an

VEB Verlag Technik
Redaktion MP
Oranienburger Str. 13/14
Berlin
1020.

MP

Programmentwicklung und -test für Einchipmikrorechner U8840/41

Johann-Georg Kretzschmar, Herbert Weber
VEB Robotron-Optima Büromaschinenwerk Erfurt

Für die Entwicklung einer neuen Generation elektronischer Schreibmaschinen wurde nach ausführlichen Voruntersuchungen auf den Einsatz eines Einchipmikrorechners U 8840 M orientiert /1/. Zur Vorbereitung von Programmentwicklung und -test waren kurzfristig die erforderlichen Voraussetzungen zu schaffen.
Dabei wurde sehr schnell klar, daß ein geeignetes Mikrorechnerentwicklungssystem nicht verfügbar ist und deshalb unter Beachtung angebotener Nachnutzungsmöglichkeiten eine eigenständige Lösung entwickelt werden mußte, die im Beitrag beschrieben wird.

1. Analyse der Anforderungen

Das Entwicklungskollektiv war bisher mit Entwicklung und Test von Programmen für den Mirkoprozessor U880 in Assemblersprache SYPS K 1520 beschäftigt und an die komfortable Unterstützung durch Mikrorechnerentwicklungssysteme MRES A 5601 gewöhnt.
Zur Sicherung einer schnellen Umstellung war daher eine möglichst große Ähnlichkeit zwischen den Entwicklungshilfsmitteln anzustreben. Als Mindestforderung galten:
- Rechnergestützte Verwaltung und Dokumentierung der Programmquellen und -phasen
- Echtzeittest der Programme mit weitgehenden Diagnose- und Eingriffsmöglichkeiten.

2. Analyse der Nachnutzungsmöglichkeiten

Begonnen wurde mit einer Untersuchung des von der Technischen Hochschule Ilmenau angebotenen Programms PAS 881 zur Quellprogrammeingabe und -assemblierung auf Basis des MC 80. Dabei stellte sich bereits nach kurzer Zeit heraus, daß dieses Programm weder bei der Quellprogrammeditierung noch bei der Assemblierung den Anforderungen entsprach. Insbesondere der Verzicht auf den Druck kommentierter Programme ist bei einer industriellen Entwicklung nicht tragbar. Auch die fehlende Unterstützung bei der Dateiarbeit ist bei Vorhaben dieses Umfanges nicht zu akzeptieren. Außerdem stand damals der inzwischen angebotene Emulator /2/, /3/ noch nicht zur Verfügung.
Danach wurde das vom VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt angebotene Programmentwicklungssystem DEUS 100 (Bürocomputer A 5120 bzw. A 5130 mit Betriebssystem UDOS) in Betracht gezogen /4/. Es enthält alle notwendigen Komponenten für die Programmentwicklung (Bildschirmeditor, Cross-Assembler, Linker einschließlich der erforderlichen Unterstützung bei der Dateiverwaltung und Dokumentation).
Störend war der durch völlig unterschiedliche Bedienung (im Vergleich zum MRES A 5601) und durch die stark abweichende Assemblersprache (im Vergleich zu SYPS K 1520 /5/) bedingte hohe Zeitaufwand für die Einarbeitung.
Die Entscheidung fiel daher zugunsten des an der Ingenieurhochschule Wismar entwikkelten Makroassemblers, der unter dem Betriebssystem MEOS V 4 auf dem MRES A 5601 läuft und eine sehr stark an SYPS K 1520 angepaßte Assemblersprache verwendet /6/. Zur Unterstützung beim Programmtest wurde kurzzeitig eine im Jahr 1979 entstandene Programmier- und Testhilfe PTH A 5602 verwendet.
Dazu war allerdings der interne Programmspeicher auf 4 KByte zu erweitern und eine on-line-Kopplung an das MRES zu realisieren. Außerdem erwies sich der Einbau einer Einzelbefehlseingabe als zweckmäßig, um Programme ab vorgegebener Adresse starten und beliebige Register des EMR laden und lesen zu können.
Auf Dauer war jedoch mit dieser Lösung eine effektive Arbeit nicht möglich. Als nächstes wurde für den Programmtest ein vom Zentrum für Forschung und Technologie des KEAW „Friedrich Ebert" zur Nachnutzung angebotener Emulatormodul in der Zusammenarbeit mit dem BC A 5120 benutzt. Die Kopplung beider Geräte über zwei serielle Schnittstellen führte jedoch häufig zu Fehlern und letztlich immer zu Abstürzen des Betriebssystems UDOS.
Deshalb und wegen der Inkompatibilität zwischen MRES und BC wurde diese Variante nicht weiterverfolgt, sondern ein an der Technischen Universität Karl-Marx-Stadt entwikkelter Emulator nachgenutzt /7/. Die Kopplung erfolgte mit dem MRES und hat sich bewährt.
Das mitgelieferte Steuerprogramm wurde allerdings vollständig überarbeitet und den eigenen Anforderungen angepaßt.
Damit entstand ein in sich geschlossenes System für Programmentwicklung und -test von EMR, das nachfolgend in einigen Aspekten ausführlicher beschrieben werden soll.

3. Editor und Assembler

Der in /6/ beschriebene Makroassembler wurde für die Übersetzung der EMR-Quellprogramme übernommen und erfüllte im wesentlichen die gestellten Anforderungen. Insbesondere bewährte sich die weitgehende Übernahme der Assemblersprache SYPS K 1520 wegen des verschwindend geringen Umstellungsaufwandes. Als nachteilig hat sich lediglich herausgestellt, daß dieser Assembler auf einer alten Version des K-1520-Assemblers aufgebaut ist, damit die erweiterte Operandenbehandlung nicht unterstützt und insbesondere keine Ausgabe der Symboltabelle zuläßt, obwohl das für den symbolischen Test vorteilhaft wäre.
Inzwischen liegt eine erweiterte und verbesserte Version dieses Assemblers als nachnutzungsfähige Lösung vor.
Als Editor muß der Free-Form-Editor des Betriebssystems MEOS benutzt werden, der die zahlreichen Arbeitserleichterungen des SYPS-K-1520-Editors und seine syntaktische Vorprüfung nicht enthält sowie die Quellprogrammerfassung und -editierung erschwert und fehleranfälliger werden läßt.

4. Emulatorelektronik

Der in /7/ beschriebene Emulator wurde übernommen und lediglich in zwei Details kritisch untersucht bzw. verbessert. Auf diese Probleme wird nachfolgend näher eingegangen.

4.1. Interruptbehandlung

Erhebliche Schwierigkeiten bereitet die Behandlung von Interruptannahmen durch den EMR während der Emulation. Einerseits sollten Eingriffe in das Interruptsystem durch die Emulation vermieden werden, um das zu testende Programm nicht zu verfälschen. So ist es beispielsweise sinnlos, ein mit Interrupts arbeitendes Echtzeitprogramm mit gesperrtem Interrupt testen zu wollen. Auch die zeitweilige Sperrung durch DI-Befehle kann das

Dipl.-Ing. Johann Georg Kretschmar (54) studierte von 1952 bis 1959 an der Friedrich-Schiller-Universität Jena und an der Technischen Hochschule Ilmenau Physik und Elektroakustik. Von 1960 an befaßte er sich in verschiedenen Funktionen im Industriezweig Datenverarbeitungs- und Büromaschinen mit Entwurf, Programmierung und Anwendung von Mikrorechnern, seit 1983 mit Forschungs- und Entwicklungsarbeiten für elektronische Schreibmaschinen.

Dipl.-Phys. Herbert Weber (36) studierte von 1970 bis 1974 an der Friedrich-Schiller-Universität Jena das Spezialgebiet Festkörper- und Tieftemperaturphysik; danach Rückkehr zum VEB Robotron-Optima Büromaschinenwerk Erfurt. Er beschäftigt sich mit Aufgaben auf dem Gebiet nichtmechanischer Druckverfahren und mit konzeptionellen Arbeiten für mikroelektronische Steuerungen elektronischer Schreibmaschinen.

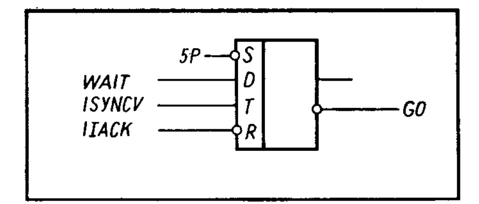


Bild 1 GO-Flip-Flop

Testergebnis verfälschen, wenn nicht durch manuelle Eingriffe (Eingabe von EI) in den jeweils interessierenden Programmabläufen der ursprüngliche Zustand des Interruptsystems wiederhergestellt wird (die automatische Behandlung ist leider nicht möglich, da das Interruptmaskenregister nicht gelesen werden kann). Andererseits können während des Programmtests Interrupts nicht uneingeschränkt zugelassen werden, da beispielsweise die Einzelbefehlsabarbeitung dann prinzipiell nur Interruptannahmen erfassen würde.

In /7/ wurde versucht, die Interruptannahme durch den EMR durch die Ausführung von DI-Befehlen unmittelbar nach dem Übergang in den Stop-Zustand zu verhindern. Das ist jedoch nicht in allen Fällen möglich und vor allem nicht automatisch rückgängig zu machen.

Es wurde deshalb ein anderer Weg beschritten: Über die Annahme von Interruptanforderungen entscheidet nur das zu testende Programm; auch im Stop-Zustand des Emulators wird jede Interruptanforderung angenommen (Bild 1). Im Stop-Zustand ist die asynchrone WAIT-Leitung aktiv, also die synchronisierte GO-Leitung passiv. Bei einer Interruptannahme wird das IACK-Signal des EMR aktiv, das Flip-Flop also gesetzt und der GO-Zustand bis zum Ende des nächsten SYNCV-Impulses (d. h. bis zum Beginn des nächsten Befehls) erzwungen.

Diese erzwungene Interruptannahme kann auf zwei Wegen erkannt werden: Die Logik für „Stop bei Interruptannahme" ist freigegeben und kann abgefragt werden, bzw. der nächste Befehl ist der erste eines Interruptbehandlungsprogramms und beginnt auf einer der sechs Vektoradressen.

Die Interruptannahme wird in jedem Fall angezeigt, und der Bediener kann dann entscheiden, ob im Interruptbehandlungsprogramm weitergearbeitet wird oder ob die Unterbrechungsanforderung zurückgesetzt und im unterbrochenen Programm fortgefahren wird (Bild 2). Im Einzelschrittbetrieb führt das bei periodischen Unterbrechungsanforderungen vom Timer allerdings dazu, daß diese Entscheidung nach jedem Befehl getroffen werden muß. In diesem Fall werden daher EI- und IRET-Befehle substituiert. Beim Übergang in den GO-Zustand muß der Bediener entscheiden, ob Interrupt freizugeben ist oder nicht. Das Steuerprogramm bietet lediglich eine Entscheidungshilfe an: Wenn eine Interruptannahme rückgesetzt bzw. ein EI- oder IRET-Befehl substituiert wurde, wird INTERRUPTFREIGABE: Y, sonst INTERRUPTFREIGABE: N zur Bestätigung angeboten (Bild 3).

4.2. Erzeugung des zentralen Steuertaktes

Der voreilende Befehlsaufruf des EMR wirkt sich bei den unterschiedlichen Befehlstypen unterschiedlich aus.

Bei Einbytebefehlen wird der nächste Befehl bereits vor dem zugehörigen $\overline{\text{SYNC}}$-Impuls adressiert (erkennbar am $\overline{\text{AS}}$-Impuls); bei Mehrbytebefehlen dagegen erfolgt die Adressierung erst nach dem $\overline{\text{SYNC}}$-Impuls (Bild 4). Über die genauen Zeitverhältnisse geben die Unterlagen /8/, /9/ keine hinreichende Auskunft, so daß eine Aufnahme der Impulsabläufe der wesentlichen Befehlstypen erforderlich war /10/.

Der zentrale Steuertakt muß gewährleisten, daß

– die Umschaltung von einer Betriebsart auf eine andere (das heißt von einem Befehlsspeicher auf einen anderen) frühestens zum Zeitpunkt t_1 (Ende $\overline{\text{DS}}$ bzw $\overline{\text{MDS}}$ des letzten Befehlsbytes des vorhergehenden Befehls)

– spätestens zum Zeitpunkt t_3 (Speicherzugriffszeit vor dem Ende von $\overline{\text{DS}}$ bzw. $\overline{\text{MDS}}$ des ersten Befehlsbytes) und

– die Übernahme der Adresse des ersten Befehlsbytes bei der Umschaltung in den Stop-Zustand frühestens zum Zeitpunkt t_2 (Ende von $\overline{\text{AS}}$)

erfolgt.

Es zeigte sich, daß die Erfüllung aller Forderungen mit keinem der vom EMR bereitgestellten Steuersignale möglich ist. Der z. B in /11/ gewählte Zeitpunkt am Ende des

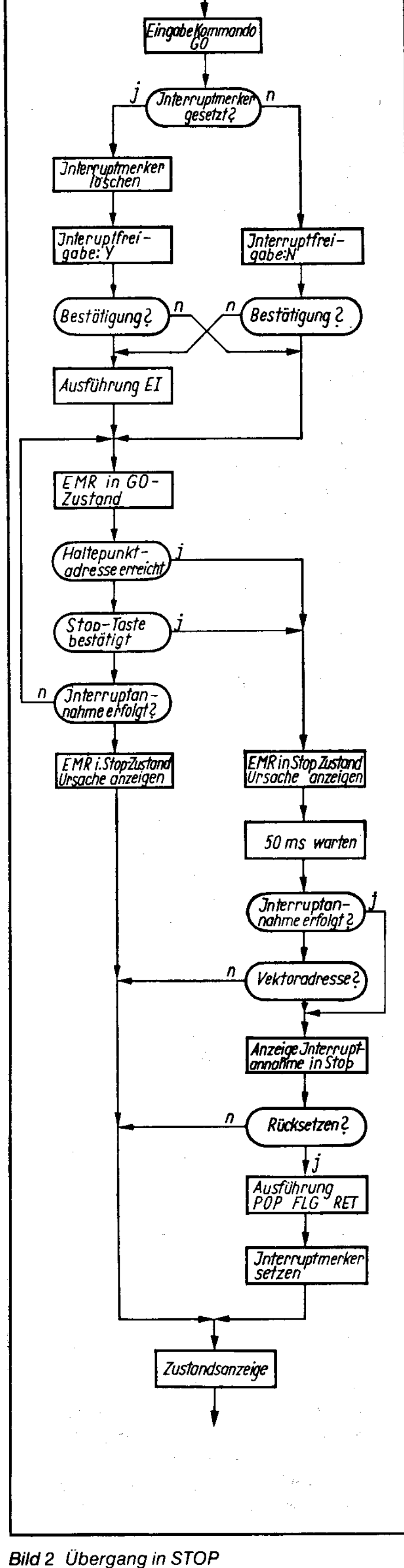


Bild 2 Übergang in STOP

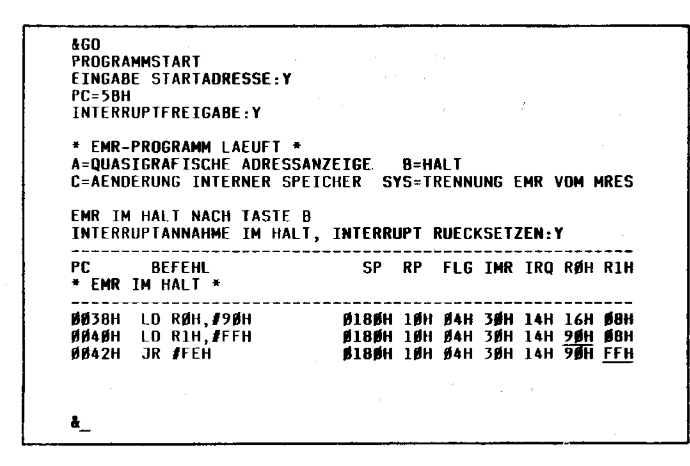
Speichererweiterung für EMR-Entwicklungsmodul

Für die Emulation von Programmen für den Einchipmikrorechner U884 wurde eine Speichererweiterung geschaffen, mit der der U881/U882-EMR-Entwicklungsmodul aus dem ZFT des KEAW nutzbar wird.

Mit der Lösung wird der interne Programmspeicher des U884 auf 4 KByte erweitert, wählbar als RAM oder EPROM (2 × U2716 bzw. 1 × U2732). Die notwendigen Modifikationen wurden auf minimalen mechanischen Eingriff und notwendige zusätzliche Bauelemente optimiert und betreffen u. a. die User-CPU, den externen Monitor-EPROM, PC-Register, Breakpoint-RAM und User-Programmspeicher. Erarbeitet wurden weiterhin ein modifiziertes Monitorprogramm und eine verbesserte, unter UDOS (MRES A 5601, BC A 5120) lauffähige Koppelsoftware mit gepufferter Terminalausgabe.

Dadurch sind z. B. das Protokollieren am Drucker, das Unterbrechen und Abbrechen der Terminalausgabe und die Arbeit am 64 × 16zeiligen Monitor möglich.

VEB Funkwerk Köpenick, Büro für Neuerwesen, Wendenschloßstraße 142-178, Berlin, 1170; Tel. 6 53 24 14.
Burkhardt



```
&GO
PROGRAMMSTART
EINGABE STARTADRESSE:Y
PC=5BH
INTERRUPTFREIGABE:Y

* EMR-PROGRAMM LAEUFT *
A=QUASIGRAFISCHE ADRESSANZEIGE   B=HALT
C=AENDERUNG INTERNER SPEICHER  SYS=TRENNUNG EMR VOM MRES

EMR IM HALT NACH TASTE B
INTERRUPTANNAHME IM HALT, INTERRUPT RUECKSETZEN:Y
----------------------------------------------------------
PC      BEFEHL                 SP   RP  FLG IMR IRQ RØH R1H
* EMR IM HALT *
----------------------------------------------------------
ØØ38H  LD RØH,#9ØH           Ø18ØH 1ØH Ø4H 3ØH 14H 16H Ø8H
ØØ4ØH  LD R1H,#FFH           Ø18ØH 1ØH Ø4H 3ØH 14H 9ØH Ø8H
ØØ42H  JR #FEH               Ø18ØH 1ØH Ø4H 3ØH 14H 9ØH FFH


&_
```

Bild 3 Bildschirmausdruck GO/STOP-Dialog

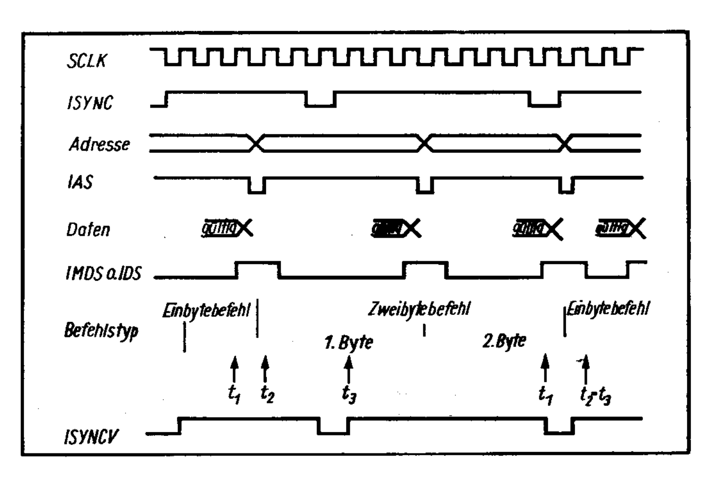


Bild 4 Schematisches Steuersignaldiagramm des EMR

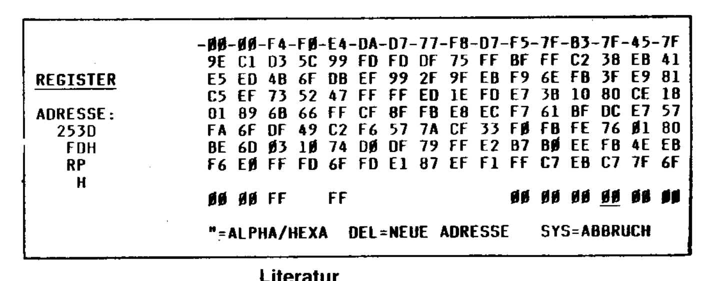
```
FUNKTIONSAUSWAHL:
RES         = EMR ZURUECKSETZEN



FUNKTIONSAUSWAHL:
RES         = EMR ZURUECKSETZEN
RSEL        = REGISTERAUSWAHL FUER ZUSTANDSANZEIGE
BRKA        = EINGABE HALTEPUNKT
BRKI        = FREIGABE HALT BEI INTERRUPT
DBRK        = HALTBEDINGUNGEN LOESCHEN (BRKA UND BRKI)
LOAD        = EMR-PROGRAMM LADEN
CD-TASTE    = EINZELBEFEHLSABARBEITUNG
NEXT        = SCHRITTWEISE PROGRAMMABARBEITUNG
GO          = PROGRAMMSTART
SAVE        = EMR-PROGRAMM RETTEN
FILL        = INTERNEN SPEICHER FUELLEN
MEM         = ANZEIGE UND AENDERUNG INTERNER SPEICHER
CM          = AENDERUNG INTERNER SPEICHER
REG         = ANZEIGE UND AENDERUNG REGISTER
CR          = AENDERUNG REGISTER
CAR         = AENDERUNG ARBEITSREGISTER
PC          = BEFEHLSZAEHLER LADEN
SP          = STAPELZEIGER LADEN
SIO, TMR, T1, PR1, TØ, PRØ, P2M, P3M, P1M, IPR,
IRQ, IMR, FLG, SPH, SPL  = STEUERREGISTER LADEN
C, Z, S, O = ANZEIGE UND AENDERUNG FLAGS

TASTE H     = FORTSETZUNG DER FUNKTIONSANZEIGE
SYS-TASTE   = ABBRUCH DER ANZEIGE
```

Bild 5 Bildschirmausdruck HELP-Funktion



```
             -ØØ-ØØ-F4-FØ-E4-DA-D7-77-F8-D7-F5-7F-B3-7F-45-7F
              9E C1 D3 5C 99 FD FD DF 75 FF BF FF C2 38 EB 41
REGISTER      E5 ED 4B 6F DB EF 99 2F 9F EB F9 6E FB 3F E9 81
              C5 EF 73 52 47 FF FF ED 1E FD E7 3B 10 80 CE 18
ADRESSE:      01 89 6B 66 FF CF 8F FB E8 EC F7 61 BF DC E7 57
  253D        FA 6F DF 49 C2 F6 57 7A CF 33 FØ FB FE 76 Ø1 80
   FDH        BE 6D Ø3 1Ø 74 DØ DF 79 FF E2 B7 BØ EE FB 4E EB
   RP         F6 EØ FF FD 6F FD E1 87 EF F1 FF C7 EB C7 7F 6F
    H
              ØØ ØØ FF    FF                ØØ ØØ ØØ ØØ ØØ ØØ

              "=ALPHA/HEXA  DEL=NEUE ADRESSE   SYS=ABBRUCH
```

Bild 6 Bildschirmausdruck REG-Funktion (Registeranzeige)

SYNC-Impulses führt mit großer Wahrscheinlichkeit zu Fehlern.
Deshalb muß ein verzögerter Steuertakt SYNCV gebildet werden, der um eine halbe Taktperiode von SCLK gegenüber SYNC verschoben ist. Zweckmäßiger als die in /7/ gewählte Verzögerung durch kapazitiv belastete Gatter wäre allerdings eine synchrone Verzögerung, gesteuert mit der Low/High-Flanke von SCLK.

5. Steuerprogrammsystem für den Emulator

Das Steuerprogrammsystem wurde vollständig neu entwickelt. Im Interesse einer weitgehenden Nutzerfreundlichkeit wurde auf komfortable Bedienerunterstützung und konsequenten Dialogbetrieb geachtet.
Einen Überblick über die Komponenten und wesentlichen Einzelheiten der Bedienung gibt der Auszug aus der HELP-Funktion (Bild 5).
Auf einige vorteilhafte Einzelheiten soll noch hingewiesen werden:
– Die symbolische Eingabe aller Operanden ist im Zusammenwirken mit dem Assembler möglich.
– Bei der Registeranzeige werden die Arbeitsregister gekennzeichnet (Bild 6).
– Bei der Lade-Funktion werden die geladenen Adreßbereiche angezeigt.
– Bei der Funktion Änderung Speicher ist die Änderung von Speicherinhalten in Echtzeit während der Programmabarbeitung möglich, um beispielsweise Beschleunigungs- und Bremskurven von Motoren im Betrieb optimieren zu können.
– Bei der EMR-Zustandsanzeige ist ein Befehls-Rückübersetzer integriert (Bild 3); der Inhalt von maximal 22 frei wählbaren Registern wird angezeigt.
– Während des Echtzeit-Programmablaufes im EMR ist eine quasigrafische Adreßanzeige einschaltbar.

6. Weiterentwicklungsmöglichkeiten

Nach nunmehr fast einjähriger Benutzung des beschriebenen Systems erscheint eine Weiterentwicklung in folgenden Richtungen sinnvoll:
– Verbesserung des Assemblers insbesondere durch Ein- und Ausgabe des Symbolverzeichnisses
– Erweiterung des Emulators durch einen über Port 0/Port 1 angeschlossenen Zusatzspeicher einschließlich Haltepunktlogik
– Erweiterung der Haltepunktlogik auf mehrere Haltepunkte
– Ausgliederung des Emulatorspeichers aus dem Adreßbereich des Grundrechners (Ansteuerung als Peripheriegerät über E/A-Kanäle).

Literatur

/1/ Weber, H.: Der Einsatz des U884 für die Steuerung elektronischer Schreibmaschinen. Vortrag zur Informationstagung Mikroelektronik, Erfurt 1986
/2/ U884-Emulatorkarte und Steuerprogrammsystem. Radio, Ferns., Elektron. 35 (1986) 10, S. 612
/3/ Roth, M.: Emulator für Einchipmikrorechner U88xx. Mikroprozessortechnik 1 (1987) 4, S. 122
/4/ Köhler, V.: UDOS-Betriebssystem für A 5120 und A 5130. In: Loll, F. (Hrsg.) Bürocomputer A 5110 - A 5120 - A 5130, Systembeschreibung Nutzererfahrungen. Verlag Die Wirtschaft, Berlin 1984
/5/ Claßen, L.; Oefler, U.: Wissensspeicher Mikrorechnerprogrammierung. VEB Verlag Technik, Berlin 1986, S. 88–148
/6/ Makro-Assembler für U881 und U882. Radio, Ferns., Elektron. 34 (1985) 1, S. 3
/7/ Franke, K.; Leichsenring, A.: Entwicklungsmodul für Einchip-Mikrorechner. Radio, Ferns., Elektron. 33 (1984) 1, S. 8
/8/ Bankel, M.: Einchip-Mikrorechner U881, U882 und U883. Radio, Ferns., Elektron. 34 (1985) 2, S. 81
/9/ TGL 42634. Unipolarer Einchipmikrorechner-Schaltkreis UB/C/D 8840 M UB/C/D 8841 M
/10/ 16-kanalige Abbildungen ausgewählter Befehlsabläufe des Z8. Unveröffentlichter Bericht
/11/ Neidhold, G.: Didaktisch gestalteter Arbeitsplatz für Einchip-Mikrorechner. Radio, Ferns., Elektron. 35 (1986) 2, S. 110
/12/ Geidel, K.-D.; Kademova-Katzarowa, P; Roth, M.: Universelle modulare Emulatoren. Radio, Ferns., Elektron. 33 (1984) 12, S. 761
/13/ Claßen, L.; Gröger, R.; Brennenstuhl, H.: Universelles Software-Entwicklungskonzept. Radio, Ferns., Elektron. 32 (1983) 8, S. 483
/14/ Kriesel, W. u. a.: Testhilfen für Einchipmikrorechner – Vergleich und Bewertung. msr 29 (1986) 3, S. 104

✉ KONTAKT ☏

VEB Robotron-Optima Büromaschinen-Werk Erfurt, FG E3, Mainzerhofplatz 13, Erfurt, 5020; Tel. 56130

ROM-Schaltkreis U2365D45 BM200 ergänzt den UB8830

Gerhard Dugnus, Siegmar Müller
VEB Mikroelektronik „Karl Marx“ Erfurt

Zur Unterstützung der Softwareentwicklung mit dem UB8830 wurde im VEB Mikroelektronik „Karl Marx“ ein ROM-Baustein U2365D45 BM200 entwickelt.
Mit dessen Hilfe ist es möglich, in einer minimalen Konfiguration Programme für kleinere Steuerungsprojekte direkt am zu steuernden Gerät zu entwickeln und zu testen.

1. Bestandteile und Zweck

Der Einsatz von Mikrorechnern in industriellen Anlagen erfordert den Test von Programmen auch direkt an der zu steuernden Einrichtung. Weil dazu oft kein Entwicklungssystem eingesetzt werden kann, soll der Rechner selbst Werkzeuge zur Testung und eventuellen Korrektur der Software besitzen. Um dem Anwender des UB8830 die Entwicklung eines kleinen, leistungsfähigen Rechners mit diesen Hilfsmitteln zu erleichtern, wurde ein ROM-Bitmuster mit der dafür erforderlichen Software entwickelt. Diese beinhaltet folgende Komponenten:
- Systemmonitor mit Assembler und Disassembler
- Editor/Debugger für Tiny-MPBASIC.

Der Systemmonitor ermöglicht in Verbindung mit dem Assembler/Disassembler die Entwicklung und den Test von Programmen in Maschinensprache, während der Editor/Debugger Entwicklung und Test von Programmen in Tiny-MPBASIC /1/ unterstützt. Weiterhin ist die serielle Kopplung mit einem Wirtsrechner möglich, so daß auch dort Programme entwickelt und EPROMs programmiert werden können.
Das ROM wird u. a. in einem Entwicklungsmodul eingesetzt, welches beim VEB Mikroelektronik „Karl Marx“ als unbestückte Leiterplatte nachgenutzt werden kann. Es handelt sich um einen universell einsetzbaren Einplatinenrechner, der einen für diese Geräteklasse sehr hohen Komfort für Programmentwicklung und -test bietet. Beim Bitmuster 200 handelt es sich um eine Arbeitsversion, die eigentlich nur zur ROM-Anprobe diente. Es erwies sich jedoch als so leistungsfähig, daß im Sinne einer breiteren und schnelleren Nutzung auf eine Überarbeitung verzichtet wurde. Deswegen erfolgen im Text Hinweise auf inzwischen bekannt gewordene Fehler und Unzulänglichkeiten.

2. Forderungen an die Hardware

Die Kommunikation mit dem Bediener erfolgt mit einem Datensichtgerät, das über eine (abgerüstete) V.24-Schnittstelle mit dem UB8830 verbunden ist. Die Übertragungsgeschwindigkeit wird beim Empfang des ersten Zeichens (Datenbit 0 dieses Zeichens muß 1 sein – z. B. %0D [Carriage Return]) durch Ausmessen des Startbits automatisch bestimmt. Die Taktfrequenz des UB8830 ist dabei unkritisch, für eine sichere Funktion bis 19200 Baud wird jedoch ein Quarz mit 7,3728 MHz empfohlen. Weiterhin kann über eine zweite serielle Schnittstelle (mit der gleichen Baudrate) ein Wirtsrechner angeschlossen werden (Bild 1a). Dieser ist als Entwicklungssystem und zum Abspeichern der entwickelten Programme nutzbar. Da der UB8830 nur einen seriellen Kanal besitzt, muß zwischen Wirtsrechner und Datensichtgerät umgeschaltet werden. Hierzu sind am Port 2 drei Steuersignale vorhanden:
- P20 = high: SIO mit Datensichtgerät verbunden (Terminal Mode)
- P21 = high: SIO mit Wirtsrechner verbunden (Host Mode)
- P22 = high: Datensichtgerät mit Wirtsrechner verbunden (Transparent Mode).

Wenn der Wirtsrechner mit einer eigenen Bildschirmsteuerung ausgestattet ist (z. B. BC A 5120, PC 1715), wird er gemäß Bild 1b mit dem UB8830-System verbunden. Die Signale an P20 bis P22 werden dann mittels Software ausgewertet.
Bild 2 zeigt die Schaltung eines Minimalsystems mit dem U2365D45 BM200. Benötigt werden neben dem UB8830 und der Anpassung an die serielle Schnittstelle noch 27 Byte RAM ab Adresse %800 und weitere ca. %100 Byte Arbeitsspeicher, die sich das ROM-Programm bei der Initialisierung von der Adresse %DFFF ab nach unten selbst sucht. Dazwischen liegt der Anwenderspeicher (auf dem Entwicklungsmodul extern erweiterbar). Die STOP-Taste (Taste nach Masse an P33) dient zum Anhalten von BASIC-Programmen und zum Verlassen des Transparent-Mode.

3. Beschreibung der ROM-Softwarekomponenten

3.1. Systemmonitor

Der Systemmonitor beinhaltet die zum Testen von Maschinenprogrammen erforderlichen Kommandos. Alle Kommandos können mit dem groß gedruckten Buchstaben abgekürzt werden.

Display ⟨adr⟩ [⟨num⟩]
Ausgabe von ⟨num⟩ Byte ab Adresse ⟨adr⟩. Wird ⟨num⟩ nicht angegeben, dann wird der Inhalt der Speicherstelle ⟨adr⟩ angezeigt und in den Eingabemodus gegangen. Danach kann entweder ein neuer Wert zum Abspeichern, RETURN zum Anzeigen des nächsten Bytes, '∧' zum Anzeigen des vorhergehenden Bytes oder 'Q' zum Verlassen des Eingabemodus eingegeben werden.

Register [⟨nr⟩[⟨wert⟩]]
Anzeigen bzw. Modifizieren von Registerinhalten. In das Register ⟨nr⟩ kann der Wert ⟨wert⟩ eingeschrieben werden. Ohne Angabe eines Wertes wird der Inhalt von Register ⟨nr⟩ angezeigt und kann wie bei Display modifiziert werden. Wurde keine Registernummer eingegeben, zeigt das Programm den aktuellen Arbeitsregistersatz.

Move ⟨adr1⟩ ⟨adr2⟩ ⟨len⟩
Kopieren von ⟨len⟩ Bytes von ⟨adr1⟩ nach ⟨adr2⟩. Move transportiert die Daten immer beginnend beim ersten Byte von ⟨adr1⟩ nach ⟨adr2⟩. Wenn ⟨adr1⟩ größer als ⟨adr2⟩ ist und beide Bereiche sich überlappen, funktioniert dieser Befehl nicht.

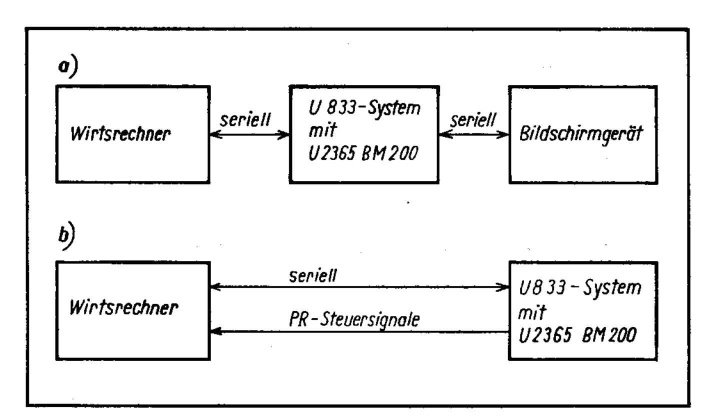


Bild 1 Kopplung mit einem Wirtsrechner
a) mit abgesetztem Terminal
b) mit eingebauter Bildschirmeinheit

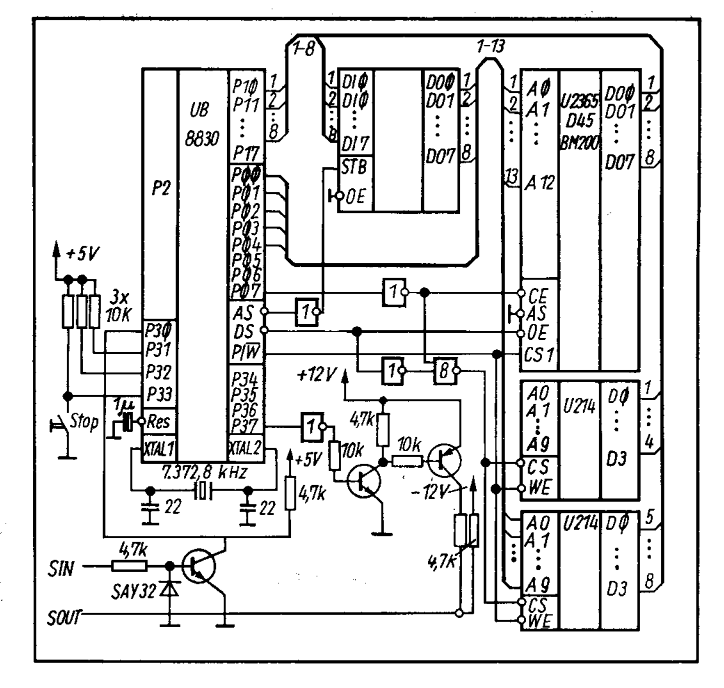


Bild 2 Minimalsystem mit U2365 D45 BM200

Compare ⟨adr1⟩ ⟨adr2⟩ ⟨len⟩
Vergleichen von ⟨len⟩ Bytes ab ⟨adr1⟩ mit ⟨len⟩ Bytes ab ⟨adr2⟩. Alle unterschiedlich belegten Speicherzellen werden angezeigt.

Jump [⟨adr⟩]
Setzen des Befehlszählers PC auf den Wert ⟨adr⟩ bzw. Anzeigen des aktuellen Wertes.

Break [⟨adr⟩]
Setzen des Unterbrechungspunktes auf Adresse ⟨adr⟩ bzw. Löschen. Der Unterbrechungspunkt darf nur auf das erste Byte eines Maschinenbefehls gesetzt werden.

Go [⟨adr⟩]
Starten des Anwenderprogramms von ⟨adr⟩ bzw. dem aktuellen Befehlszählerstand.
Das ROM-Programm enthält einige Hardwareinitialisierungen, die beim Start mit RESET ins Anwenderprogramm entfallen. Ein typisches Beispiel ist die Zeichenausgabe über die SIO. Das Bit 4 im IRQ-Register wird nach RESET in den Nicht-bereit-(0)-Zustand gesetzt. Man muß daher entweder mit OR IRQ, #%10 dieses Bit setzen oder die erste Ausgabe ohne Abfrage dieses Bits ausführen.

Execute [⟨adr⟩]
Ausführen eines Unterprogramms, dessen Eintrittspunkt der aktuelle Befehlszählerstand bzw. ⟨adr⟩ ist. Der Unterbrechungspunkt wird nicht gesetzt, und die Anwenderregisterspeicher werden weder gesetzt noch gerettet.

Next [⟨nr⟩]
Ausführen der nächsten ⟨nr⟩ Programmschritte. Ohne Argument wird der nächste Maschinenbefehl ausgeführt. Bei Extended Memory Timing muß das Register %F8 mit dem R-Befehl des Monitors auf %92 eingestellt werden. Damit wird auf Normaltiming umgestellt.
Dieser Befehl arbeitet mit dem Timer-O-Interrupt. Bei IO-Interrupts im Anwenderprogramm kann das zu Fehlfunktionen führen.

Transparent
Übergang in den Transparent Mode (s. oben). Der Transparent Mode wird mit dem Drücken der STOP-Taste verlassen.

Load ⟨file⟩
Laden eines Programms mit dem Namen ⟨file⟩ vom Wirtsrechner im Tektronix-Format mit Quittung (s. Punkt 4.).

Upload ⟨file⟩ ⟨adr⟩ ⟨len⟩ [⟨entry⟩]
Abladen eines ab ⟨adr⟩ gespeicherten Programms (bzw. Daten) mit der Länge ⟨len⟩ Bytes auf dem Wirtsrechner, ebenfalls im Tektronix-Format. Mit ⟨entry⟩ kann ein Eintrittspunkt angegeben werden, der ansonsten wie ⟨adr⟩ gesetzt wird. Beim Laden des Programms wird der Anwenderbefehlszähler mit diesem Wert geladen.

eXterne Speicheransprache
Setzt man vor eines der Kommandos Compare, Display oder Move ein 'X', so wirken diese Programme im Datenspeicher. Move transportiert dann Bytes vom Daten- in den Programmspeicher. Bei 'XD' (eXternal Display) können keine Daten in den Datenspeicher eingegeben werden.

Im ROM-Programm werden die Register 4... %2F benutzt. Die Anwenderregister dieses Bereichs sind deshalb als virtuelle CPU im RAM gespeichert und werden beim Programmstart mit 'Go' an die richtige Stelle gebracht. Dasselbe trifft auf die Steuerregister IMR, FLAGS, RP, SPH und SPL zu.
Der MPBASIC-Editorteil benutzt die Register %78 bis %7F.
MPBASIC-Programmzeilennummern, die in Hexschreibweise ein %0D im Lowbyte haben, werden nicht richtig einsortiert. Man vermeidet deshalb ungerade Zeilennummern oder prüft die Zahl mit dem PRINTHEX-Befehl.
Die Adressen %800 bis %81A werden vom ROM initialisiert. Nach RESET müssen eventuell geänderte Zellen in diesem Bereich neu eingestellt werden. Der Editor setzt die Eingabe (z. B.)

PROC SETR %50, 0, PTC %40
in
PROC SETR %50, 0; PROC PTC %40

um, um einen internen Fehler des UB8830 zu umgehen.

3.2. Assembler und Disassembler

Mit Assemble ⟨adr⟩ kann ein bei ⟨adr⟩ beginnendes Programmstück kontrolliert bzw. eingegeben werden. Zuerst wird der Disassembler aufgerufen und der bei ⟨adr⟩ stehende Befehl in disassemblierter Form angezeigt. Dahinter kann ein neuer Befehl in mnemonischer Form eingegeben werden. Mit CR (ASCII Carriage Return) wird der nächste Befehl rückübersetzt. Mit 'Q' kommt man in den Systemmonitor zurück. Der Assembler versteht neben den üblichen U8810-Mnemoniks die folgenden Pseudo-Befehle:

BVAL ⟨byte⟩
Byte Value, Definieren eines 8-Bit-Wertes

WVAL ⟨word⟩
Word Value, Definieren eines 16-Bit-Wertes

DEFS ⟨len⟩
Define Storage, Freihalten von Speicher der Länge ⟨len⟩

DEFM '⟨text⟩'
Define Message, Einsetzen von ASCII-Text in das Programm.

8-Bit-Operanden werden dezimal oder hexadezimal eingegeben, Hexzahlen werden mit einem vorangestellten '%' gekennzeichnet. 16-Bit-Operanden sind immer hexadezimal ('%' kann dabei entfallen). Die Adressen von Arbeitsregistern oder Arbeitsdoppelregistern sind immer dezimal einzugeben. 'a' (Commercial at) kennzeichnet direkte Operanden. Die Indexregister bei indizierter Adressierung werden in runde Klammern eingeschlossen.

3.3. Editor/Debugger für Tiny MPBASIC

Mit # als Monitorkommando wird Tiny-MPBASIC (im folgenden TMPB abgekürzt) aufgerufen. Es meldet sich der Editor/Debugger mit seinem Prompt '#'. Steht noch kein Programm im Speicher, muß zuerst das Kommando NEW gegeben werden. Beim Einschalten wird ein eventuell im gepufferten CMOS-RAM stehendes Programm nicht gelöscht. Daher ist der RAM beim erstmaligen Einschalten undefiniert belegt, und es fehlt die Endekennung. Diese wird mit 'NEW' initialisiert.

Die erste Adresse des BASIC-Textes steht im Doppelregister 6,7. Sie wird vom ROM-Programm mit %900 initialisiert, kann aber bei Bedarf mit PROC SETRR [6,% ⟨neue Adr⟩ I] im MPBASIC verändert werden. Letzteres trifft auch auf die Adresse der Prozedurtabelle zu, die im Register 8,9 steht und mit %0000 initialisiert wird (bedeutet: keine Prozedurtabelle vorhanden). Weitere Kommandos in dieser Programmkomponente sind:

LIST [⟨zeile⟩]
Auflisten der spezifizierten Programmzeile bzw. des gesamten Programms. Wenn nur eine Zeile aufgelistet wurde, kann durch anschließendes CR die nächste Zeile gelistet werden. Letztere Betriebsart ist insbesondere für den Betrieb mit einzeiligen Datensichtgeräten gedacht, wie sie bei den vorgesehenen Einsatzfällen typisch sind.

RUN
Starten des BASIC-Programms. RUN ruft den MPBASIC-Interpreter im UB8830 auf. Dieser benutzt für PRINT, PRINTHEX und INPUT die Programme PUTCHR und GETCHR /1/ mit den Eintrittspunkten %815 bzw. %818 (im externen Speicher). Das ROM-Programm initialisiert diese Adressen mit Sprungbefehlen in die ROM-eigenen Routinen. Bei der Verwendung anderer Ein- und Ausgabemittel können diese Sprungadressen mit dem Assembler auf andere Werte eingestellt werden.

Im Falle eines Fehlers bei der Abarbeitung des BASIC-Programms wird mit einer Fehlermeldung angehalten. Die folgenden Fehler werden dabei erkannt:

#1 Überlauf des GOSUB-Stack
#2 Auftreten von RETURN ohne GOSUB
#3 Auftreten von GOSUB ohne RETURN (Meldung erst am Programmende)
#4 Division durch Null
#8 Zahlenbereichsüberschreitung.

Bei Zahlenbereichsüberschreitung wird die Programmabarbeitung im Falle von Addition und Subtraktion nicht gestoppt. Die Fehlermeldung erfolgt dann erst am Programmende. Dadurch wird verhindert, daß eine gewollte Überschreitung, die z. B. bei hexadezimaler Adressenarithmetik auftreten kann, ein Anhalten des Programms bewirkt.

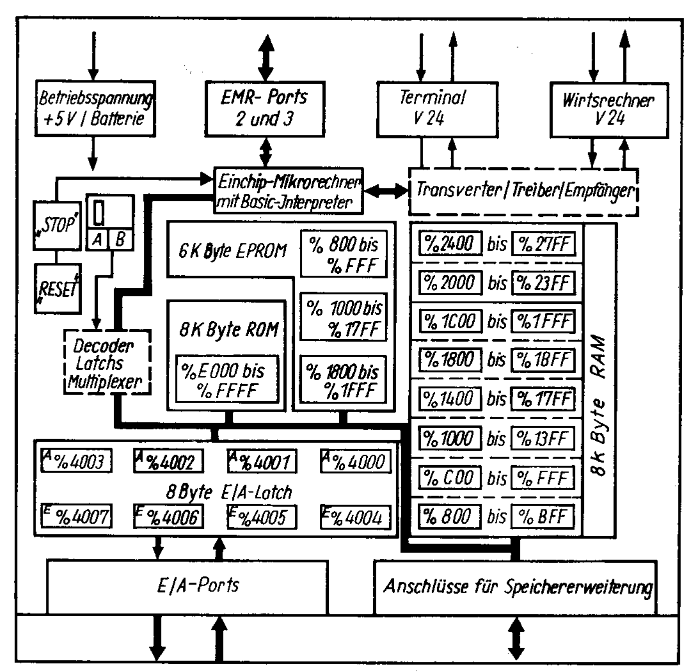
EXEC
Starten eines BASIC-Programms, jedoch kein STOP bei Fehlern

CONT[⟨zeile⟩]
Fortsetzen des Programms ab ⟨zeile⟩ bzw. ohne Argument nach STOP

STEP[⟨zeile⟩]
Abarbeiten der angegebenen bzw. der nächsten Programmzeile. Wenn einmal STEP ausgeführt wurde, kann mit CR die Folgezeile abgearbeitet werden.

SIZE
Ausgabe folgender Daten von Programm und Prozedurtabelle:
Anfangsadresse, Endadresse und Länge

GET ⟨programmname⟩
Laden eines BASIC-Programms vom Wirtsrechner. Das Programm wird auf die Adresse zurückgeladen, von der es auf den Wirtsrechner abgeladen wurde.



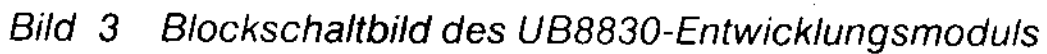


Bild 3 Blockschaltbild des UB8830-Entwicklungsmoduls

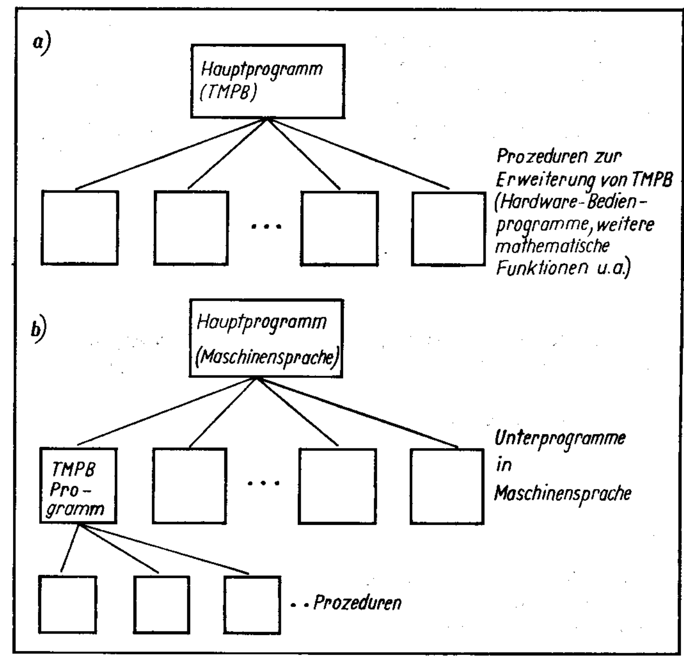


Bild 6 Mögliche Strukturen von Programmen

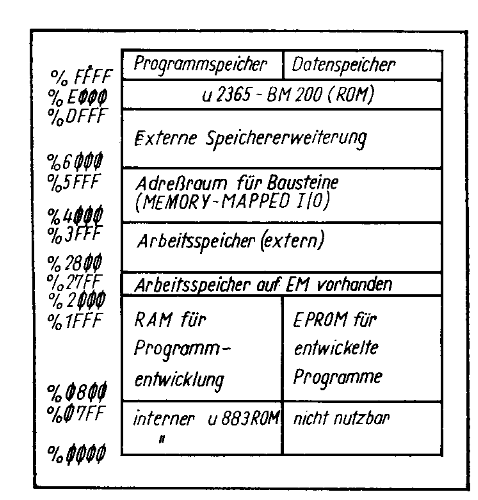


Bild 4 Speicheraufteilung im U8830-Entwicklungsmodul

SAVE ⟨programmname⟩
Abladen eines BASIC-Programms auf den Wirtsrechner

BYE
Rückkehr in den Systemmonitor.

4. Datenaustausch mit einem Wirtsrechner

Bevor der Datenaustausch beginnt, wird das Kommando "LOAD ⟨filename⟩" (Load oder GET) oder das Kommando "SEND ⟨filename⟩" (bei Upload und SAVE) an den Wirtsrechner gegeben, damit dieser das entsprechende, als Kommando aufrufbare Programm LOAD oder SEND in seinen Arbeitsspeicher laden und starten kann. Wenn SEND gestartet wurde, darf der Wirtsrechner auf die zu ihm gesendeten Zeichen kein Echo mehr ausgeben. Benutzt wird das Tektronix-Format zur seriellen Übertragung. Die Daten werden in Blöcke geteilt, von denen jeder eine Startadresse, die Bytezahl, zwei Testsummen sowie die eigentlichen Daten enthält.

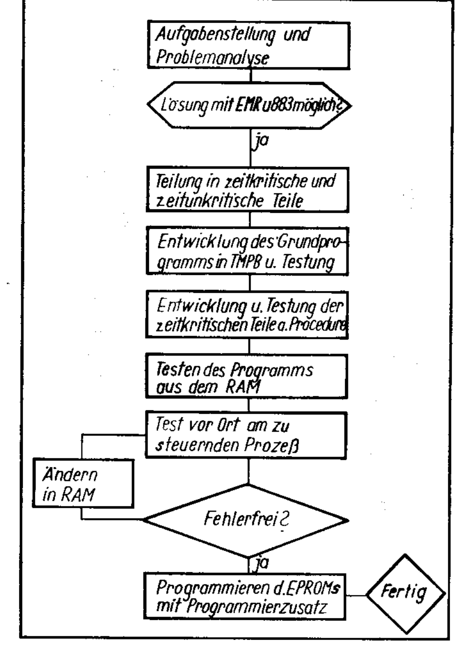


Bild 5 Programmentwicklung mit dem U8830-Entwicklungsmodul

Diese Blöcke sind folgendermaßen aufgebaut:
/⟨adresse(4)⟩ ⟨bytezahl (2)⟩ ⟨testsumme1 (2)⟩ ⟨Datenbyte(2)⟩ ...
... ⟨Datenbyte(2)⟩ ⟨testsumme2(2)⟩ ⟨CR⟩
Dabei markiert '/' den Anfang eines Datenblocks oder einer Fehlermeldung.
⟨adresse(4)⟩ ist die Adresse für das erste Datenbyte (in 4 ASCII Zeichen dargestellt).
⟨bytezahl(2)⟩ gibt die Anzahl der Datenbytes an.
⟨testsumme1(2)⟩ wird aus Anfangsadresse und Bytezahl gebildet.
⟨datenbyte(2)⟩ ist ein Datenbyte im Tektronix-Format.
⟨testsumme(2)⟩ wird für die Datenbytes berechnet.
⟨CR⟩ ist das ASCII-Zeichen CR (%0D).

Die Kodierung des Tektronix-Formats ist so festgelegt, daß ein Halbbyte (eine Hexziffer) als ASCII-Code übertragen wird (z. B. %8B wird als Folge %38 %42 codiert). Der letzte Block beinhaltet die Eintrittsadresse:

/⟨eintritt(4)⟩ ⟨bytezahl(2)⟩ ⟨testsumme(2)⟩ ⟨CR⟩

Anstelle der Blockanfangsadresse steht der Eintrittspunkt; die Bytezahl ist immer Null (%30 %30), und die Daten einschließlich testsumme2 fehlen.
Nach dem Senden eines Datenblocks wird die Quittung von der Gegenstelle erwartet. Bei fehlerfreier Übertragung wird mit '0' (%30) quittiert, '7' bedeutet Testsummenfehler, eine '9' bedeutet Systemfehler im Wirtsrechner. Bei Testsummenfehlern wird die Übertragung bis zu 10mal wiederholt, um Störungen zu unterdrücken. Auch eine Fehlermeldung der Form

//⟨fehlermeldung als ASCII-Text⟩ ⟨CR⟩

führt zum sofortigen Abbruch der Datenübertragung, wobei das zweite '/'-Zeichen den Block als Fehlermeldung kennzeichnet.

5. UB8830-Entwicklungsmodul

Bild 3 zeigt das Blockschaltbild des UB8830-Entwicklungsmoduls (im folgenden EM abgekürzt) und Bild 4 die Speicheraufteilung. Es kann ab Adresse %800 mit bis zu 6 KByte Daten- und 6 KByte Programmspeicher bestückt werden (RAMs 1K×4 und EPROM U2716). Darüber hinaus stehen maximal 2 KByte Arbeitsspeicher ab Adresse %2000 zur Verfügung. Mit S1 kann der 6-KByte-RAM-Bereich wahlweise als Daten- oder Programmspeicher geschaltet werden. Der EPROM ist der jeweils alternative Speicherbereich. Um mit dem ROM U2365D45 BM200 arbeiten zu können, muß der RAM als Programmspeicher geschaltet sein.
Mit dem Kommando "XM" des Systemmonitors ist es möglich, den EPROM-Inhalt (gewöhnlich Programme) in den RAM-Bereich auf die gleichen Adressen zu kopieren. Dies vereinfacht das Testen und Bearbeiten mit

dem Monitor. Damit ist es ebenfalls möglich, das Programm aus dem RAM direkt über einen Programmierzusatz in EPROMs zu programmieren. Mit dem Schalter S2 kann der RAM ein Schreibverbot erhalten, wodurch der EMR UB8830 bei der Startroutine einen ROM auf Adresse %800 ff. vermutet, was wiederum zum Start des Programms ab %812 führt. Damit kann das Programm so starten, als ob es bereits in einem EPROM stünde.
Von den maximal 2 KByte Arbeits-RAM werden vom ROM-Programm etwa %100 Byte benötigt. Der Rest dieses Bereichs steht dem Anwender zur Verfügung. Die Umschaltung zwischen Programm- und Datenspeicher betrifft nur die 6 KByte Entwicklungsspeicher, also nicht den ROM-Bereich ab %E000 und den Arbeits-RAM. Alle RAM-Bausteine sind in eine gepufferte Betriebsspannung angeschlossen, die es z. B. erlaubt, ein an der zu steuernden Anlage geändertes und getestetes Programm an einem anderen Ort (etwa im Labor) im EPROM zu programmieren oder auf einem Massenspeicher abzulegen. Acht Latches stehen mit je 8 Bit als Memory-mapped-I/O zur Verfügung, davon 4 für Eingabe- und 3 für Ausgabezwecke, wobei jeweils die Output-Enable- bzw. Strobeeingänge der Latches mit herausgeführt sind.
Der interne getriebene Bus des EM steht an einem 58poligen Steckverbinder für Erweiterungen zur Verfügung. Der Adreßraum des EM ist voll dekodiert, für jeweils 8 KByte liegen sogenannte Blockenablesignale am Erweiterungsstecker an.
Ein Transverter erzeugt die für die abgerüsteten V.24-Schnittstellen benötigten Spannungen von +12 V und −12 V. Das gesamte EM läßt sich daher mit einer Betriebsspannung von 5 V (ca. 1,6 A) betreiben. Mit dem zusätzlich verfügbaren EPROM-Programmierzusatz stellt das EM bereits ein für kleinere Anwendungsfälle komplettes Entwicklungssystem dar. Da die Anforderungen für Tastatur und Anzeige sehr unterschiedlich sind und oft bereits ein Rechner mit diesen Einheiten und einer seriellen Schnittstelle zur Verfügung steht, ist hier keine Standardlösung vorgesehen. Bekannt sind Lösungen mit Bildschirmsteuerung (komplettes Terminal), aber auch mit LED-Kombinationen wie VQC10 oder VQB201. Im allgemeinen wird dazu ein zweiter EMR zur Steuerung verwendet.

6. Softwareentwicklung

Im Bild 5 ist der Ablauf der Programmentwicklung mit dem UB8830-EM dargestellt. Vor Ort können Programmkorrekturen ohne Wirtsrechner durchgeführt werden. Einzige Voraussetzung ist eine seriell betriebene Tastatur- und Anzeigeeinheit. Ein typisches Programm für den UB8830 ist in Tiny-MPBASIC geschrieben. Nur zeitkritische Teile und in MPBASIC zu umständliche Teile schreibt man in Maschinensprache und bindet sie als Prozeduren in das MPBASIC-Programm mit ein (Bild 6a). Da es möglich ist, ein MPBASIC-Programm wie ein Maschinenprogramm aufzurufen (siehe /1/), kann ein Programmaufbau nach Bild 6b gewählt werden. Hinweise und Beispiele zu Tiny-MPBASIC sind in /3/ enthalten.

(Literatur und Kontakthinweis auf Seite 251)

Interpretativ arbeitende speicherprogrammierbare Steuerung SKS 02

Dr. Ronald Schoop, Holger Lisson,
Prof. Dr. Wolfgang Weller
Humboldt-Universität zu Berlin,
Sektion Elektronik

1. Einleitung

Eine zunehmende Ablösung von verbindungsprogrammierten Steuerungen, wie z. B. Relais oder ursalog 4000, auch für Steuerungsprobleme geringerer Komplexität wird wesentlich durch die Entwicklung neuer speicherprogrammierbarer Steuerungen (SPS), wie etwa EFE 700 /1/ oder S 2000 S /2/, forciert. Zur Lösung von Steuerungsaufgaben mit einer Anzahl von weniger als 40 Ein- und Ausgängen fehlen jedoch aufwandsminimale und damit wirtschaftliche SPS, insbesondere wenn auch der notwendige Aufwand für Programmier- und Inbetriebnahmegeräte mit berücksichtigt wird.
Um auch für derartige Steuerungsprobleme die bekannten Vorteile von SPS /3/ nutzbar zu machen, wurde eine speicherprogrammierbare Kleinsteuerung SKS 02 auf der Basis von Einchipmikrorechnern (EMR) mit folgender Zielstellung entwickelt /4/ bis /14/:

- minimaler Aufwand für Steuer- und Programmiergerät bei Aufgaben mit weniger als 30 Ein- und Ausgängen
- einfache Handhabung bei Programmierung und Inbetriebnahme
- hohe Störsicherheit
- Kompatibilität zum System ursalog 4000.

Die letztgenannte Forderung ermöglicht neben dem Aufbau eigenständiger SKS auch eine Aufwertung bisheriger Steuerungen auf der Basis von ursalog-4000-Baugruppen.

2. Gesamtkonzept

Zur Programmierung und Inbetriebnahme werden der Programmiergerätekomplex (PGK) und der Steuergerätekomplex (SGK) miteinander seriell (UART; 19,2 kbit/s) entsprechend Bild 1 verbunden. Im Steuerungsbetrieb kann diese Schnittstelle zur Kopplung mit einer zweiten Steuerung oder einem übergeordneten Rechner genutzt werden, wobei Merkerbelegungen oder Programme übertragen werden können.

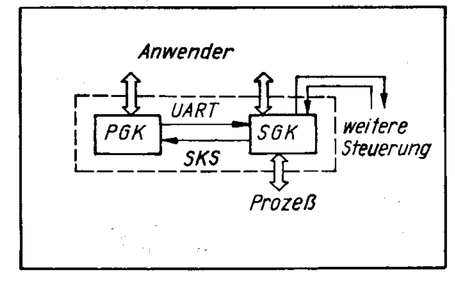


Bild 1 Gesamtstruktur der SKS 02

3. Struktur des Steuergerätekomplexes

Die programmierte SKS kann mit dem Modell des endlichen Automaten beschrieben werden, der den Steueralgorithmus A verwirklicht. Notwendige Zähl- und Zeitfunktionen werden dabei gesonderten Funktionsgruppen Z bzw. T (Bild 2) zugeordnet. Demzufolge brauchen diese im zu programmierenden Steueralgorithmus A' nur insoweit berücksichtigt werden, als sie über Eingänge X' (Zählkonstante erreicht bzw. Zeit abgelaufen) und Ausgänge Y' (Zählerstand erhöhen bzw. Zeitablauf starten) verkoppelt sind. Zur Erhöhung der Flexibilität werden zusätzlich die Zähl- bzw. Zeitkonstanten vom Steueralgorithmus A', d. h. vom Anwenderprogramm festgelegt, so daß nichtüberlappende Mehrfachnutzungen prinzipiell möglich sind.
Sämtliche Funktionsgruppen T, Z und A' sind im Steuergerät der SKS 02 implementiert.
Die Grundkonfiguration wird dabei aus einer Zentraleinheit (ZE) und einer Expanderkarte (EX) gebildet. Zur Aufrüstung kann diese Anordnung um eine weitere Expanderkarte gemäß Bild 3 erweitert werden.
Den Kern der Zentraleinheit (Bild 4) bildet der Einchipmikrorechner UB 8820. Das Betriebssystem ist in einem EPROM U 2716 und das Anwenderprogramm wahlweise in einem 2-KByte-EPROM oder RAM abgelegt, was einer Kapazität von ca. 600 ... 900 Anwenderbefehlen entspricht. Über das serielle Interface kann eine TTL-kompatible Schnittstelle angeschlossen werden. Eine Funktionsgruppe zur Anzeige erlaubt das Darstellen der Belegung von 8 internen Merkervariablen und, in Verbindung mit der watch-dog-Schaltung, die Anzeige eines Fehlerkodes im Störfall. Letzterer wird durch eine Autodiagnose (EMR, RAM, EPROM) erkannt oder durch einen undefinierten Rechner-„absturz" verursacht und führt zum Stopp des Programmlaufes und zum Rücksetzen der Ausgänge. Weiterhin wird ein selbsttätiges Anlaufen nach Spannungsausfall und -wiederkehr verhindert.
In der Zentraleinheit implementiert sind 8 Zähler/Zeitgeber mit den Bereichen 1 .. 256 bzw. 0,1 ... 25,6 s, die mit einem Anwenderbefehl parametriert werden. Es können 128 Zustandsvariablen (Merker) belegt werden. Die Arbeitsweise der Zentraleinheit ist in

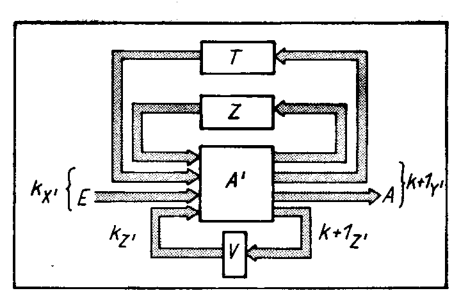


Bild 2 Partiell dekomponiertes Automatenmodell

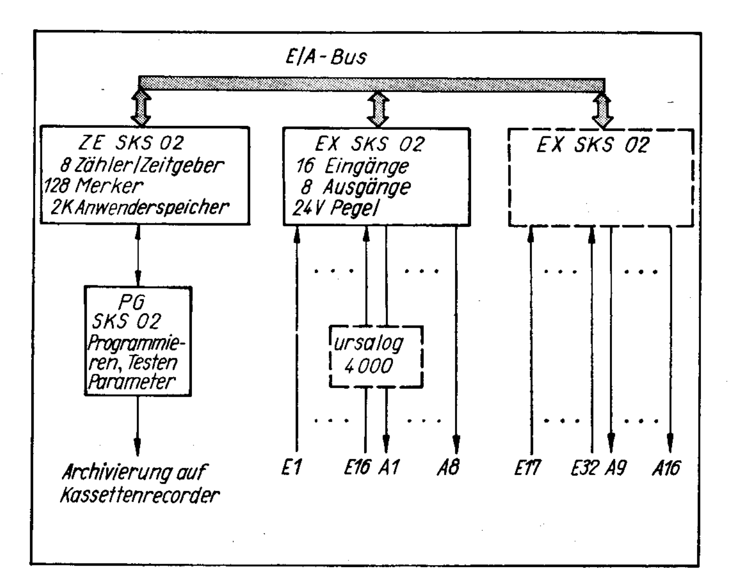


Bild 3 Struktur des Steuergerätes SE 221/222

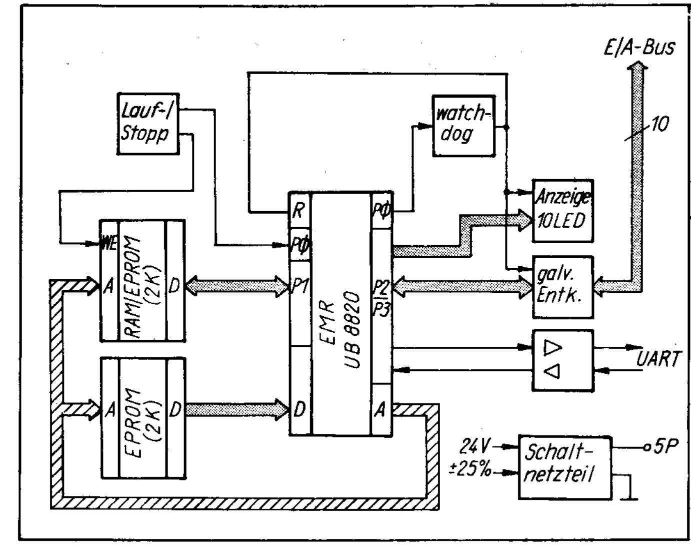


Bild 4 Blockschaltbild der Zentraleinheit

Abhängigkeit von der Programmierung zyklisch oder prozeßgesteuert, wobei in prozeßbedingten Pausen eine erweiterte Autodiagnose vorgenommen wird. Darüber hinaus kann über 2 Eingänge ein spezielles, vom Anwender definiertes Interruptprogramm gestartet werden, was z. B. für die Realisierung einer Not-Aus-Routine von Vorteil ist.

Die Anpassung des von der Zentraleinheit gebildeten E/A-Busses an die Prozeßein- und -ausgänge erfolgt durch die Expanderkarte. Die Bauelementebasis dieser Karte bilden eine CMOS-Logik und der Schaltkreis D 410, der eine störsichere bzw. kurzschlußfeste Ausführung der prozeßseitigen 16 Ein- und 8 Ausgänge gewährleistet. Der Datenaustausch mit der ZE erfolgt über Multiplexer bzw. Demultiplexer und eine Busanpaßschaltung. Eine Taktung der Eingaberegister erfolgt bei jedem Zustandsübergang über ein spezielles Steuersignal, wodurch eine korrekte Ermittlung der Übergangsbedingungen ermöglicht wird. Dieses Signal wird selbsttätig aus dem Programm und somit aus dem Steueralgorithmus abgeleitet, und zwar beim Programmablaufgraphen von der Operation und bei Booleschen Gleichungen vom Programmende.

Die Stromversorgung erfolgt ausgehend von 24 V ± 25% über ein Schaltnetzteil.

Tafel 1 Befehlssatz

Befehl	Operation	Operand					Folgeadresse	sonst.
		/E,E	/M,M	/A,A	/T,T	/Z,Z		
SETZE	SE		•	•	•	•		
WARTE	WA	•	•		•	•		
FRAGE	FR	•	•		•	•	MARKE	
GEHE	GE						MARKE	
LADE	LA	•	•		•	•		
UND	UN	•	•		•	•		
ODER	OD	•	•		•	•		
GLEICH	GL		•	•	•	•		
PAR	PA				•	•		Zeit- bzw. Zählkonstante
SENDE	SD		•					

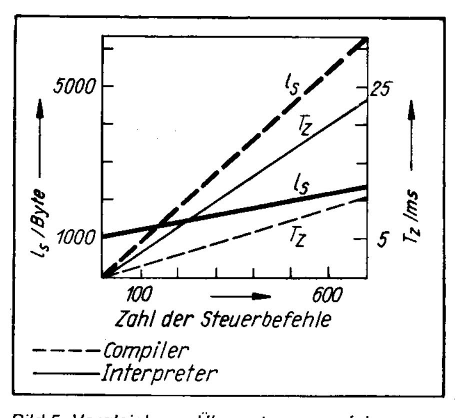
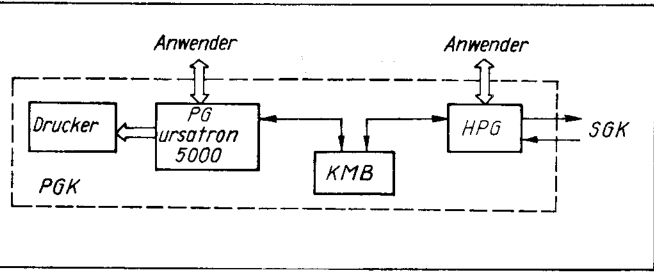


Bild 5 Vergleich von Übersetzungsverfahren

Bild 6 ▼ Struktur des Programmiergerätekomplexes



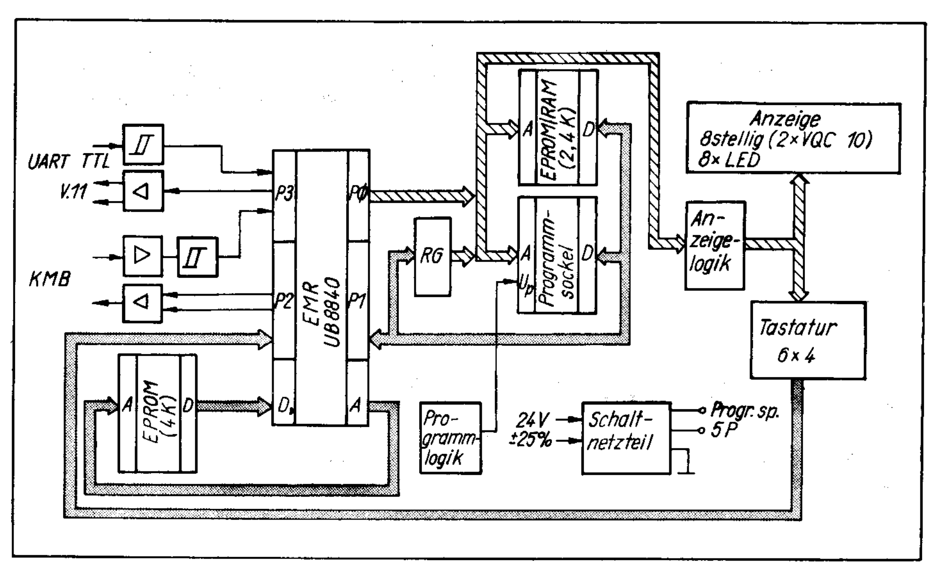


Bild 7 ▲ Blockschaltbild des Programmiergerätes PG 222

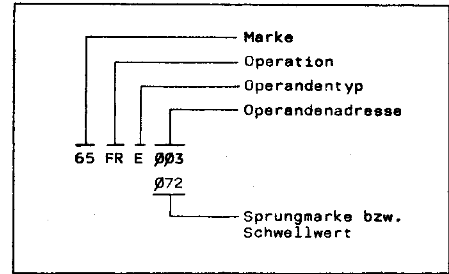


Bild 8 Struktur der Steuersprachbefehle

4. Übersetzungsprinzip

Als Verfahren zur Übersetzung aus der Programmier- in die Maschinensprache der SKS existieren folgende Möglichkeiten:

- Assembler
- Compiler
- Translator/Interpreter.

Die Verwendung eines Assemblers kommt aufgrund der ungenügenden Problemorientiertheit der Programmiersprache nicht in Betracht.
Einen Vergleich zwischen Compiler und Translator/Interpreter mit den Parametern Speicherplatzbedarf l_s im Steuergerät und Zustandsübergangszeit T_z zeigt Bild 5 /5/. Dementsprechend stehen sich ein größerer Hardwareaufwand bei Verwendung eines Compilers einerseits und höhere Zustandsübergangszeiten bei Einsatz eines Translators/Interpreters andererseits gegenüber. Für die Implementierung der SKS 02 wurde das letztgenannte Prinzip verwendet, da erstens bereits ab etwa 200 Steuersprachbefehlen der Hardwareaufwand für das Steuergerät geringer ist, zweitens die Zustandsübergangszeiten von etwa 50 ms/1000 Steuersprachbefehlen noch vertretbar sind und drittens mit dem Interpreterkode eine transparente Schnittstelle für eine Programmarchivierung oder Programmierung auf verschiedenen Rechnern (z. B. BC A 5120, PG ursatron 5000) existiert.

5. Programmiergerätekomplex

Folgende Betriebsarten werden gefordert:

- Programmerstellung
- Programmtest
- Programmarchivierung u. -dokumentation.

Die Struktur des Programmiergerätekomplexes zur Erfüllung dieser Funktionen ist in Bild 6 dargestellt. Zur Erzielung eines minimalen Grundaufwandes werden alle zur Programmierung und Inbetriebnahme notwendigen Funktionen in einem Handprogrammiergerät (HPG) konzentriert. Die zusätzlichen Funktionen Archivieren und Dokumentieren werden über ein Interface zu handelsüblichen Kassettenmagnetbandrecordern (KMB) realisiert. Für das verwendete Aufzeichnungsformat existieren Dienstprogramme auf dem Programmiergerät ursatron 5000 (für den BC A 5120 in Vorbereitung), die das

- Editieren
- Übersetzen
- Rückübersetzen
- EPROM-Programmieren und
- Archivieren

von Anwenderprogrammen ermöglichen.

☒ KONTAKT ☎

Humboldt-Universität zu Berlin,
Sektion Elektronik, Wissenschaftsbereich 3 – Automatisierungstechnik,
Invalidenstr. 110, Berlin, 1040;
Tel. 2803563
Dr.-Ing. R. Schoop

Das HPG erlaubt neben diesen Teilfunktionen zusätzlich:

- Step- und Breakpoint-Betrieb
- Anzeige von Bitakkuinhalt und Merkerbelegung sowie
- Fehlersuchlauf.

Ein Blockschaltbild des Handprogrammiergerätes PG 222 zeigt Bild 7. Den Kern der Lösung bilden der EMR UB8840 (oder UB8820) und ein Betriebssystemspeicher von 4-(bzw. 2×2-)KByte-EPROM. Die Bedienung erfolgt über 24 Tasten (Funktionstastatur mit Doppelbelegung), die Anzeige ist mittels eines 8stelligen Punkt-Matrix-Displays für Befehle und den Anwenderdialog sowie mittels 8 LED für die ausgewählte Betriebsart ausgeführt. Im PG 222 stehen Schnittstellen zur EPROM-Programmierung, zum KMB und ein UART-Kanal mit TTL- und V.11-kompatiblem Pegel zur Verfügung. Die Stromversorgung erfolgt ausgehend von 24 V±25% über ein Schaltnetzteil.

6. Programmierung

Als Beschreibungsgrundlage zur Programmierung können strukturorientierte Beschreibungsformen wie Logikplan, Kontaktplan oder Boolesche Gleichungen dienen. Daneben erweist es sich als vorteilhaft, auch funktionsorientierte Beschreibungen, wie den Programmablaufgraphen (PAG) /15/, zuzulassen. Diese entstehen beim systematischen Entwurf bereits zu einem sehr frühen Zeitpunkt /16/ und erlauben – in Verbindung mit einer bei der SKS 02 angewendeten möglichen nichtzyklischen Programmabarbeitung – minimale Zustandsübergangszeiten im Millisekundenbereich, da nur relevante Eingangsvariablen verarbeitet werden müssen.
Die Struktur der 10 konzipierten Steuersprachbefehle ist in Bild 8 und der gesamte Befehlssatz in Tafel 1 dargestellt. Die ersten 4 Befehle dienen zur Programmierung auf der Grundlage von PAGen und können eindeutig den Elementen des PAG zugeordnet werden. Die nachfolgenden vier Befehle erlauben die Programmerstellung auf der Basis von Booleschen Gleichungen bzw. Kontaktplänen. Der Parametrierbefehl weist einem Zähler bzw. Zeitgeber einen Schwellwert zu. Der Sendebefehl bewirkt einen Austausch der Belegung beliebiger Merker zur Kopplung von 2 Steuerungen. Die Syntax für eine gemischte Anwendung der 3 Befehlsmengen ist in Bild 9 angegeben.
Beispiele zur Programmierung sind in Bild 10 dargestellt.

7. Implementierung

Das Steuergerät SE 221 besteht aus 3 Leiterplatten (Zentraleinheit, Expander, Frontplatte) und befindet sich in einem Einschub (170 × 98 × 40 mm³). Das Programmiergerät PG 222 wird aus 3 Leiterplatten (Tastatur/Anzeige, Rechnerteil, Schaltnetzteil) gebildet und ist in einem handlichen Plastgehäuse (220 × 110 × 50 mm³) untergebracht.
Eine Dokumentation (Hard- und Softwarebeschreibung, Schaltungsunterlagen, Betriebssysteme, Inbetriebnahmesoftware und Bedienungsanleitung) und Leiterplatten (in beschränkter Anzahl) können angeboten werden.

Dr.-Ing. Ronald Schoop (29) studierte an der Humboldt-Universität zu Berlin, Sektion Elektronik, von 1976 bis 1981; Diplomabschluß auf dem Fachgebiet Automatisierungstechnik. Im Anschluß an eine Tätigkeit als wissenschaftlicher Assistent an derselben Einrichtung erfolgte im Jahre 1985 die Promotion A zu Problemen speicherprogrammierbarer Steuerungen. Seit 1985 ist er arbeitsteilig im VEB Secura Berlin für die Entwicklung von Gerätesteuerungen verantwortlich und an der Humboldt-Universität tätig.

Dipl.-Ing. Holger Lisson (26) studierte an der Humboldt-Universität zu Berlin, Sektion Elektronik, von 1982 bis 1986. Nach seinem Diplomabschluß auf dem Fachgebiet Automatisierungstechnik ist er als Forschungsstudent an derselben Einrichtung tätig. Arbeitsgebiet speicherprogrammierbare Steuerungen.

Prof. Dr. sc. techn. Wolfgang Weller (52) studierte von 1953 bis 1959 an der Technischen Hochschule Dresden, Fachgebiet Regelungstechnik. Von 1953 bis 1966 war er als Entwicklungsingenieur, Gruppen- und Abteilungsleiter im Institut für Regelungstechnik Berlin tätig. Nach seiner Promotion A an der Technischen Universität Dresden im Jahre 1966 war er bis 1967 Dozent am Higher Institute of Electronics Menouf (ARÄ) und Berater am Ministry of Higher Education in Kairo. Im Anschluß an seine Tätigkeit als Bereichsleiter für Systemgrundlagen im Institut für Regelungstechnik Berlin und als nebenamtlicher Hochschullehrer an der Wilhelm-Pieck-Universität Rostock erfolgte 1970 die Berufung zum Ordinarius für Technische Kybernetik an der Humboldt-Universität zu Berlin, wo er die Funktion eines Wissenschaftsbereichsleiters ausübt. 1973 habilitierte er an der Wilhelm-Pieck-Universität Rostock.

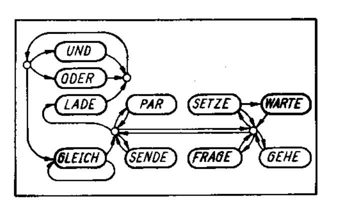


Bild 9 Syntax der Anwenderprogramme

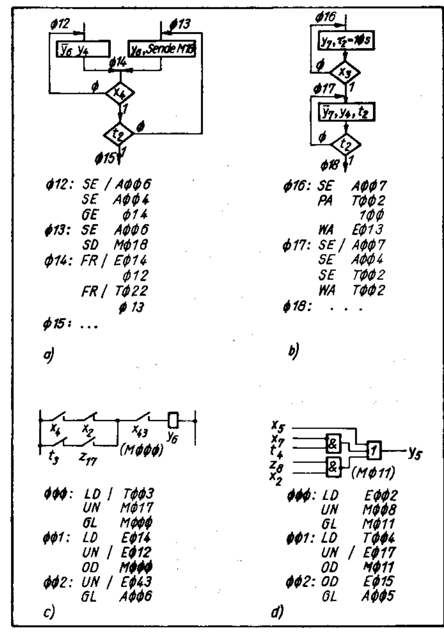


Bild 10 Programmierbeispiele

(Fortsetzung auf S. 238)

Was ist eine RISC-Architektur?

Konventionelle Prozessor-Konzeptionen und -Architekturen befassen sich in erster Linie mit dem Problem, das Zusammenspiel zwischen einem schnellen Prozessor und dem verhältnismäßig langsamen Arbeitsspeicher so zu gestalten, daß der langsame Zugriff auf die Programmbefehle im Arbeitsspeicher die Verarbeitungsgeschwindigkeit des Prozessors nicht zu stark vergrößert. Deswegen wurden komplexe, leistungsstarke Befehle vorgesehen, die dann im Prozessor als sogenannte Mikrocoderoutinen ablaufen und dabei in der Regel viele Prozessorzyklen erfordern. Mit dieser Technik, der Complex Instruction Set Computer (CISC), gelang es, zeitaufwendige Speicherzugriffe auf das wirklich notwendige Maß zu reduzieren und dabei die Geschwindigkeit des Prozessors voll auszunutzen.

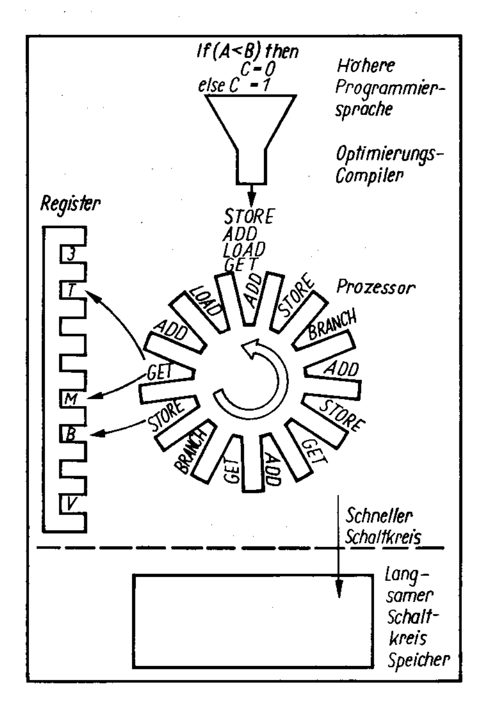
Nun geht die Entwicklung einerseits dahin, daß auch die Zugriffszeiten zu den Speichern immer kürzer und so die Fähigkeiten mächtiger Befehle oft nicht voll genutzt werden. Die Maschine betreibt intern viel „Leerlauf". Andererseits ergaben Untersuchungen verschiedener Computerhersteller, daß für viele Anwendungen die meisten Computer den Großteil ihrer Rechenzeit zur Ausführung von nur 20 Prozent ihres Befehlssatzes verwenden. Aus diesen Gründen versuchte man, mit anderen Architekturen einen Ausweg zu finden. Eine solche Lösung scheint sich mit der RISC-Architektur anzubieten. Ein Computer mit RISC-Architektur (Reduced Instruction Set Computer) kommt mit wesentlich eingeschränktem Vorrat an Maschinenbefehlen aus. In Verbindung mit einem sehr leistungsfähigen Compiler wird die hohe interne Verarbeitungsgeschwindigkeit realisiert. Die Zugriffszeiten zu modernsten Arbeitsspeichern entsprachen etwa einem Prozessorzyklus. Somit kann während der Abarbeitung eines Befehls der Folgebefehl aus dem Arbeitsspeicher geladen werden. Der Instruktionssatz eines RISC-Prozessors umfaßt nur einfache Befehle, die möglichst während eines einzigen Prozessorzyklus ausgeführt werden können. Fast alle Befehle sind Register-Register-Operationen, was der Verarbeitungsgeschwindigkeit zugute kommt. Der laufzeitoptimierte Objektcode wird von RISC-Compilern u. a. dadurch erzeugt, daß Operanden möglichst lange in Registern gehalten und Speicherzugriffe nur dann ausgeführt werden, wenn neue Operanden benötigt oder nicht mehr gebrauchte abgespeichert werden müssen.

Ein weiteres typisches Merkmal ist die pipelineartig-überlappende Befehlsabarbeitung, die ebenfalls zur Geschwindigkeitserhöhung beiträgt (siehe Bild).

Eine Reihe internationaler Computerhersteller – IBM, Hewlett-Packard, Nixdorf, GEI und Inmos – haben bereits RISC-Maschinen entwickelt. So wird beispielsweise von IBM das System 6150 angeboten. Der 32-Bit-Mikroprozessor des Mikrocomputersystems enthält mehr als 50000 Transistorfunktionen und ist als Einzelchip in RISC-Architektur ausgeführt. Der MP verfügt über nur 118 Befehle in 2- und 4-Byte-Format, die zum größten Teil innerhalb eines einzigen Verarbeitungszyklus ablaufen.

Die Verarbeitungsgeschwindigkeit beträgt zwischen 1,6 und 2,1 MIPS bei einer Zugriffszeit von rund 170 Nanosekunden. Noch wesentlich interessanter dürfte die an der Universität von Kalifornien Berkeley (USA) entwickelte RISC-Maschine sein. Nur 30 Maschinenbefehle bilden den Instruktionssatz. Bestimmte Programme liefen auf diesem Computer mehr als doppelt so schnell wie auf einem anderen zum Vergleich herangezogenen Mikrocomputer mit CISC-Prozessor.

Dennoch stellt die RISC-Architektur nicht das Nonplusultra dar. Als Nachteil führen Kritiker des RISC-Konzepts u. a. an, daß das Aufstellen eines typischen RISC-Befehlssatzes nur



auf statistischer Grundlage basiert, denn es werden nur die Befehle implementiert, die besonders häufig benötigt werden. Damit sind diese Computer für bestimmte Anwendungen sehr schnell, für viele andere aber eben auch zu spezialisiert. So werden die universelleren CISC noch lange Zeit ihre Berechtigung haben.

I. P.

Literatur

/1/ RISC – die schnelle Architektur. information + berichte + modelle, Wien (1987)5, S. 19

/2/ RISC: Verkaufsargument oder technischer Durchbruch? Elektronik (1986) 9, S. 189–190

(Fortsetzung von S. 237)

Literatur

/1/ Weißbach, G.; Schröter, K.: Zentrale Verarbeitungseinheit in der programmierbaren Kleinsteuerung EFE 700. Radio, Ferns., Elektron., Berlin 35 (1986) 5

/2/ Firmenschrift KEAW: Speicherprogrammierbare Steuereinrichtung S 2000 S

/3/ Weller, W.; Wilke, H.: Programmierbare Steuereinrichtungen. Band 195 der REIHE AUTOMATISIERUNGSTECHNIK. VEB Verlag Technik, Berlin 1981

/4/ Weller, W.: PAPS-Konzeption einer im Echtzeitbetrieb arbeitenden PSE. Forschungsbericht AT 80-2. Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1980

/5/ Schoop, R.: Entwurf einer Komplexlösung für eine speicherprogrammierbare Kleinsteuerung auf der Basis von Einchipmikrorechnern. Dissertation A, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1984

/6/ Löbe, D.: Softwarezähler bzw. Zeitgeber. Großer Beleg, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1984

/7/ Günther, Th.: Magnetbandkassettenabspeicherung. Großer Beleg, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1984

/8/ Lisson, H.: Programmierung der SKS 02 auf dem PG ursatron 5000. Ingenieurbeleg, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1986

/9/ Kolbe, U.: Modifikation der Zentraleinheit der SKS 01 mittels Expander und Erprobung. Ingenieurbeleg, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1985

/10/ Kolbe, U.: Software für ein Taschenprogrammiergerät. Diplomarbeit, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1987

/11/ Petersen, R.: Hardware für ein Taschenprogrammiergerät. Ingenieurbeleg, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1987

/12/ Schoop, R.; Lisson, H.: Speicherprogrammierbare Steuerung SKS 02-Steuergerät SE 221/222. Forschungsbericht, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1987

/13/ Schoop, R; Lisson, H.: Speicherprogrammierbare Steuerung SKS 02- Handprogrammiergerät PG 222. Forschungsbericht, Humboldt-Universität zu Berlin, Sektion Elektronik, Berlin 1987

/14/ Schoop, R.: Modulares System zum Aufbau programmierbarer Kleinsteuerungen auf der Basis von Einchipmikrorechnern. 13. Arbeitstagung „Entwurf von Schaltsystemen, Systementwurf". ZKI, Dresden 1984

/15/ Killenberg, H.: Verhaltensbeschreibung von Schaltsystemen mit Hilfe von Programmablaufgraphen. messen steuern regeln, Berlin 19 (1976) 6

/16/ Killenberg, H.; Krapp, M.; Flurschütz, K.: Struktureller Entwurf industrieller Steuerungen. Entwurfsrichtlinie, VEB Werkzeugmaschinenkombinat „7. Oktober" Berlin

REDABAS

Arbeit mit Datenbanken (Teil I)

Dr. Thilo Weller, Matthias Donner
Karl-Marx-Universität Leipzig, Sektion Wirtschaftswissenschaften/Organisations- und Rechenzentrum

Mit dieser Beitragsfolge sollen Anregungen zur Programmierung in REDABAS vermittelt werden, die über die reine Dialogarbeit weit hinaus gehen. Gleichzeitig kommt es den Autoren darauf an, zu zeigen, daß das Standardsoftwarepaket REDABAS zahlreiche Schnittstellen zu anderen Programmen bietet, die bei der entsprechenden Aufgabenstellung ausgeschöpft werden sollten.

1. Einführung

Mit REDABAS (kompatibel zu dBASE II) steht ein universell einsetzbares RElationales DAtenBAnkbetriebsSystem für Personalcomputer/Arbeitsplatzcomputer zur Verfügung, das mit einem einheitlichen Sprachkonzept sowohl den Aufbau und die Verwaltung von Datenbeständen als auch Befehle für eine eigene Anwendungsprogrammierung bereithält /1/.
Vor allem bei komplexen Problemen kann der Bereich der Datenerfassung, Datenänderung und der Datenverknüpfung günstig mittels der integrierten Programmiersprache bearbeitet werden. Die Anwendung der REDABAS-Programme führt in den meisten Fällen zu einer Arbeitserleichterung, z. B. für Nutzer ohne rechentechnische Spezialausbildung.
Sowohl die Sprachsyntax als auch das Datenbanksystem selbst erlauben dabei eine modulare Gestaltung von Programmen, wodurch eine stufenweise Programmerstellung ermöglicht wird.

2. Syntaxnotation

Für die Notation der REDABAS-*Anweisungen* werden verwendet:
GROSSBUCHSTABEN beinhalten REDABAS-Schlüsselwörter, die genau so geschrieben werden müssen.
[...] Eckige Klammern beinhalten einen Teil der Anweisung, der wahlweise angegeben werden kann.
⟨...⟩ Kleingeschriebenes in spitzen Klammern beinhaltet einen Teil der Anweisung, der durch den Nutzer auszufüllen ist.
Für die Schreibweise der *Programmbeispiele* wird keine gesonderte Notation festgelegt.

3. Aufbau von REDABAS-Programmen

Der REDABAS-Interpreter ermöglicht also
– die Arbeit auf einer Anweisungsebene: zur Abarbeitung einmaliger Aufgabenstellungen, zum Testen von Datenbanken u. a.
– die Abarbeitung von Befehlsdateien (Programmen) mittels der integrierten Programmiersprache: mit der einfachen Möglichkeit, Anweisungen zusammenzufassen, durch einmaliges Programmieren wiederholte Nutzungsmöglichkeiten zu schaffen.
Die Programmiersprache beinhaltet zum einen die Verwendung der Anweisungen zur Datenbankarbeit auf der Anweisungsebene sowie weitere Programmstrukturelemente. Damit können z. B. solche Anweisungen auch direkt in der Programmierung genutzt werden wie zum

– Erstellen von Dateien
CREATE
COPY u. a.
– Erfassen von Daten
APPEND u. a.
– Ändern von Dateiinhalten
BROWSE
EDIT
DELETE
PACK u. a.
– Anzeigen von Dateiinhalten
DISPLAY
LIST u. a.

Zur *Erstellung* von REDABAS-Programmen kann ein Texteditor, z. B. das Textverarbeitungsprogramm TP (Option N: Bearbeiten einer Programmdatei) oder der intergrierte Programmeditor selbst genutzt werden.

4. Erstellen von Befehlsdateien mit dem integrierten Programmeditor und Aufruf von Befehlsdateien

Das *Aktivieren des Programmeditors* zur formatfreien Erstellung und Modifizierung von Anweisungsfolgen sowie zur Speicherung in einer Befehlsdatei erfolgt mit

MODIFY COMMAND ⟨programmname⟩

Nach Eingabe dieser Anweisung schaltet REDABAS in einen Eingabemodus um, wofür die vorhandenen CTRL-Tasten-Kombinationen zu verwenden sind; zum Beispiel:

CTRL – W für Speichern des Programms auf Diskette mit der Dateierweiterung. PRG (vollständiger Dateiname/max. 8 Zeichen: programmname.PRG) und Verlassen des Programmeditors
CTRL – Q für das Abbrechen der Programmbearbeitung und Verlassen des Editors *ohne* Speichern.

Zum Zeitpunkt der Programmerstellung erfolgt keinerlei Prüfung der eingegebenen Anweisungen, erst bei der Abarbeitung der Befehlsdatei!
Wichtig ist die Beachtung der Leistungsgrenzen des Programmeditors:
– maximale Länge einer Befehlszeile 77 Zeichen (Byte)
– maximale Größe einer Befehlsdatei ca. 4000 Zeichen
– keine Textsuch- sowie Blockverschiebungsanweisungen möglich.
Mittels
DO ⟨programmname⟩
kann nun der *Aufruf und die Abarbeitung* der Befehlsdatei ⟨programmname⟩. PRG erfolgen.

Die Aufrufmöglichkeiten bestehen sowohl aus der Anweisungsebene heraus (d. h. nach dem Promptzeichen „:") als auch aus einer anderen Befehlsdatei (d. h. Verschachtelungen von Befehlsdateien in Form einer Unterprogrammtechnik sind möglich). Die mögliche Schachtelungstiefe ist durch die maximale Anzahl von 16 gleichzeitig eröffneten Dateien begrenzt, wobei jede DO-Anweisung das Eröffnen einer entsprechenden (weiteren) Befehlsdatei bewirkt. Außerdem sind hier auch die anderen momentan eröffneten Dateien zu berücksichtigen (z. B. Datenbank-, Index-, Reportdateien)! Eine Befehlsdatei kann auch *direkt aus der Betriebssystemebene* heraus aufgerufen werden:

A > REDABAS ⟨programmname⟩

Die *Ausführung* einer Befehlsdatei *wird beendet*, wenn entweder das Ende der Datei oder die Anweisung

RETURN

erreicht wird.

5. Programmierte Datenein- und -ausgabe

5.1. Programmierte Dateneingabe

Die Eingabemöglichkeiten über die Konsole umfassen die

– ACCEPT-Anweisung
zur *Zeichenketteneingabe* für Speichervariable
– INPUT-Anweisung
zur Eingabe von *Daten beliebigen Typs* für Speichervariable
– READ-Anweisung
zur Eingabe von *Daten beliebigen Typs* über eine *Eingabemaske*, die zuvor mit a ... GET definiert wurde
– WAIT-Anweisung
zur Eingabe eines *Zeichens* beliebigen Typs nach einer Programmunterbrechung.
REDABAS unterscheidet also zwischen *Feldvariablen* (Diese bestimmen alle zu einer Datenbankstruktur gehörenden Datensatzelemente mit einem vom jeweiligen Datensatz abhängigen Dateninhalt.) sowie *Speichervariablen*. (Diese existieren unabhängig von der jeweils bearbeiteten Datenbank, maximale Länge jeweils 254 Byte, maximale Zahl 64.)

ACCEPT [⟨zeichenkette⟩] TO ⟨speichervariable⟩

Diese Anweisung ermöglicht die Eingabe beliebiger *alphanumerischer* Ausdrücke von der Tastatur in eine Speichervariable. Die Eingabe braucht nicht in Anführungszeichen eingeschlossen zu werden; ACCEPT faßt jede Eingabe als alphanumerischen Wert auf.
Vor der Eingabe kann eine Erläuterung (Zeichenkette) über die Art der zu erwartenden Daten gegeben werden, die in Begrenzungszeichen (Anführungszeichen ", Hochkommas ' oder eckige Klammern []) einzuschließen ist.
Die Speichervariable muß vor der Ausführung der Anweisung nicht definiert werden, sondern wird automatisch erzeugt.

Beispiele:
ACCEPT 'Programm wiederholen?' (Ja/Nein)
TO ANTWORT
ACCEPT 'Welcher Datensatz?' TO DS

Die Ausgabe der ACCEPT-Anweisung läßt sich nicht auf eine bestimmte Bildschirmposition festlegen (immer ab Spalte 0; Vorsicht: Bildschirmmenüs können leicht überschrieben werden!).

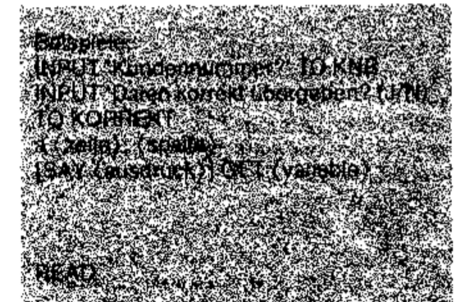
INPUT [⟨zeichenkette⟩
TO ⟨speichervariable⟩

Die Arbeitsweise mit der INPUT-Anweisung ähnelt der bei der ACCEPT-Anweisung. Unterschiede bestehen darin, daß Daten *beliebigen* Typs eingegeben werden können; der Typ der Speichervariablen bestimmt sich durch die Art der Eingabe:

■ Numerische Werte werden numerischen Speichervariablen zugewiesen.

■ Alphanumerische Werte (hier bei der Eingabe in Begrenzungszeichen einzuschließen!) werden alphanumerischen Speichervariablen zugewiesen.

■ Bei J (oder T) bzw. N (oder F) erfolgt die Zuweisung des entsprechenden logischen Wertes TRUE bzw. FALSE an eine logische Speichervariable.


READ

Diese mächtigste Anweisung zum Einlesen von Daten bei der Programmabarbeitung beinhaltet die *Positionierung der Ausgabe* eines Ausdrucks auf Bildschirm (bzw. Drukker) *sowie die Eingabe von Daten beliebigen Typs* in die bezeichnete Variable.
Üblicherweise gelten folgende Positionsangaben: zeile 0...23, spalte 0...79.
Mit der SAY-Option kann ein Ausdruck (Verknüpfung von Konstanten, Variablen, Funktionen durch in REDABAS zulässige Operatoren) beliebigen Datentyps ausgegeben werden.
GET ermöglicht die Eingabe von Daten in die betreffende Variable, wobei zuerst deren momentaner Inhalt angezeigt wird.
Diese Variable *muß* vorher definiert sein (siehe auch STORE-Anweisung). Die Variable darf sowohl eine Speichervariable als auch eine (Datenbank-) Feldvariable sein.
Das eigentliche Einlesen der Daten (und deren Zuweisung zur Variablen) an den durch GET auf Bildschirm markierten Positionen erfolgt erst mit der READ-Anweisung.

Beispiel:
STORE 0.00 TO BETRAG
@ 5,10 SAY 'Betrag' GET BETRAG
READ
Die vollständige Syntax der @...SAY...GET-Anweisung enthält zwei weitere Optionen:

@ ⟨zeile⟩, ⟨spalte⟩
[SAY ⟨ausdruck⟩ [USING 'maske']]
[GET ⟨variable⟩ [PICTURE 'maske']]
Maske beinhaltet ein oder mehrere Maskenzeichen.

Mit der USING-Option wird festgelegt, in welcher Form die im Ausdruck stehenden Daten ausgegeben werden sollen, analog setzt die PICTURE-Option bestimmte Eingabeformate fest.

Maskenzeichen	Bedeutung für USING	Bedeutung für PICTURE
# oder 9	Ausgabe einer Ziffer des Ausdrucks (bzw. Leerzeichens, wenn an dieser Stelle keine Ziffer)	erlaubt nur Eingabe einer Ziffer sowie von "+", "-", ",", "."
X	Ausgabe eines beliebigen ASCII-Zeichens	erlaubt Eingabe eines ASCII-Zeichens
A	–	erlaubt nur Eingabe eines Buchstabens
& oder *	Ausgabe einer Ziffer oder Ausgabe von "&" bzw. "*" anstelle einer führenden Null	–
!	–	Umwandlung von Klein- in Großbuchstaben

Die Standardeinstellung für die behandelte Anweisung ist SET FORMAT TO SCREEN (Bildschirmein-, ausgabe); bei Druckerausgabe ist SET FORMAT TO PRINT zu verwenden.

WAIT [TO ⟨speichervariable⟩

WAIT veranlaßt REDABAS, das laufende Programm zu unterbrechen, seine Fortsetzung erfolgt mit der Eingabe eines beliebigen Zeichens von der Konsole. Dieses Zeichen kann wahlweise in der Speichervariablen gespeichert werden, um es für die Programmsteuerung zu nutzen.

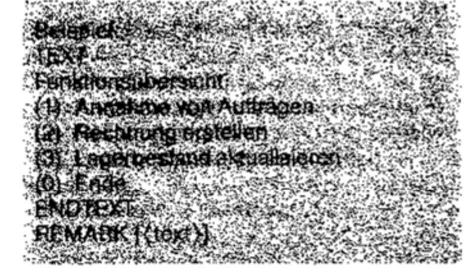


5.2. Programmierte Datenausgabe

Ausgabemöglichkeiten umfassen u. a. die
– a... SAY-Anweisung
zur Ausgabe von Ausdrücken an bestimmten Positionen (siehe Pkt. 5.1, ohne GET-Option)
– Text... ENDTEXT-Anweisung
zur *einfachen* Ausgabe von (z. T. umfangreichen) Kommentaren
– REMARK-Anweisung
zur weiteren Ausgabe kurzer Kommentare während der Abarbeitung von Befehlsdateien
– ERASE-Anweisung
zum Löschen des Bildschirms.

TEXT
.
.
.
⟨text⟩
.
.
.
ENDTEXT

Mit dieser Anweisungsstruktur können bei der Ausführung von Befehlsdateien eine oder mehrere Kommentarzeilen *ohne Positionsangaben* auf einfache Weise ausgegeben werden. TEXT und ENDTEXT müssen auf separaten Zeilen stehen. Der Ausgabetext erfolgt am Bildschirm *unverändert*.


(2) Rechnung erstellen
(0) Ende
ENDTEXT

Die Anweisung dient der Anzeige eines (kurzen) Kommentartextes während der Programmdurchführung am Bildschirm, insbesondere zur Information des Bedieners.

ERASE

löscht den Bildschirm und positioniert den Cursor an die 1. Bildschirmposition. Wird die @-Anweisung zur Bildschirmausgabe verwendet, entaktiviert ERASE alle GET-Optionen.

5.3. Bemerkungen

Mit den im Pkt. 5 behandelten Anweisungen ist eine Vielzahl komfortabler Möglichkeiten der Online-Verarbeitung gegeben. Diese Anweisungen sollten u. a. für Menügestaltung, programmierte Datenerfassung und -aktualisierung genutzt werden.

6. Grundstrukturen der strukturierten Programmierung

6.1. Folge

Die Folge (Sequenz) beinhaltet, daß Anweisungen nacheinander abgearbeitet werden (Bild 1).
Es empfiehlt sich folgende Vorgehensweise bei der Erstellung der übergeordneten Befehlsdatei:

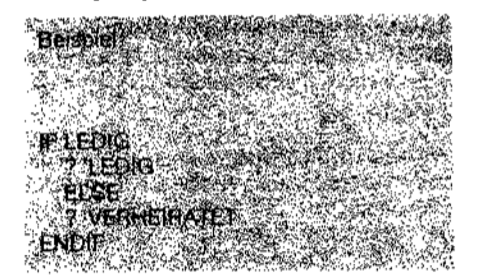
anweisung _ 1:
NOTE ⟨text⟩ bzw. *⟨text⟩ (Kommentar als Programmkopf; wird bei der Abarbeitung übergangen)
anweisung _ 2:
ERASE (Löschen des Bildschirms und Positionierung des Cursors in der linken oberen Ecke)
anweisung _ 3:
SET TALK OFF (Unterdrücken der Systemmeldungen)

6.2. Verzweigung

Die Verzweigung beinhaltet in Abhängigkeit von einer Bedingung die alternative Durchführung verschiedener Anweisungen.

6.2.1. Alternative

Mittels der IF-Anweisung wird in Abhängigkeit von der Erfüllung einer Bedingung (Ausdruck, der als Auswahlkriterium dient) die eine oder die andere Anweisung (bzw. Anweisungsfolge) ausgeführt (Bild 2).



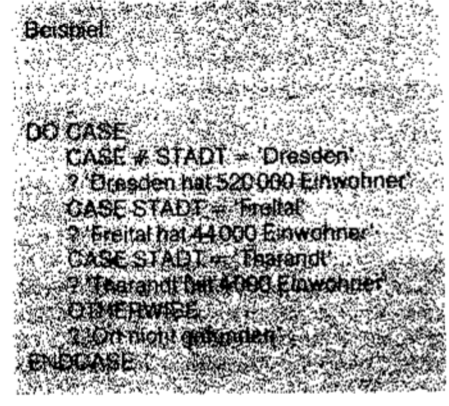
```
Beispiel:

IF LEDIG
   ? 'LEDIG'
   ELSE
   ? 'VERHEIRATET'
ENDIF
```

Ist die Bedingung nach der IF-Anweisung erfüllt (wahr), werden die unmittelbar folgenden Anweisungen ausgeführt, anderenfalls die Anweisungen nach ELSE. Enthält eine IF-Anweisung keinen ELSE-Zweig (unvollständige Alternative), werden bei Nichterfüllung die Bedingung aller Anweisungen bis zur nächsten ENDIF-Anweisung übergangen.
Zwischen IF und ENDIF können z. B. weitere IF-Anweisungen stehen, die Schachtelungstiefe ist nicht begrenzt, sollte aber übersichtlich gehalten werden.

6.2.2. Fallunterscheidung

Mittels der CASE-Anweisung erfolgt die alternative Auswahl aus verschiedenen Anweisungen (Anweisungsfolgen) entsprechend den aufgeführten Bedingungen (Bild 3).



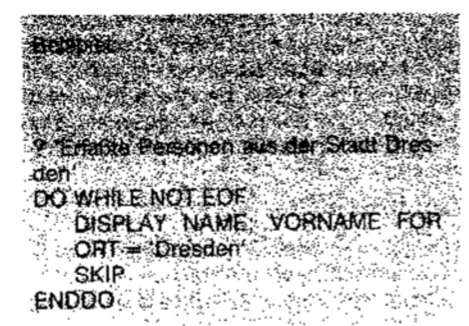
```
Beispiel:

DO CASE
   CASE STADT = 'Dresden'
   ? 'Dresden hat 520000 Einwohner'
   CASE STADT = 'Freital'
   ? 'Freital hat 44000 Einwohner'
   CASE STADT = 'Tharandt'
   ? 'Tharandt hat 4000 Einwohner'
   OTHERWISE
   ? 'Ort nicht gefunden'
ENDCASE
```

Die Bedingungen werden nacheinander geprüft. Sobald eine Bedingung zutrifft, wird die zugehörige Anweisung bzw. Anweisungsfolge ausgeführt, und der CASE-Block ist damit abgearbeitet. Enthält CASE einen OTHERWISE-Zweig, wird, falls keine Bedingung zutrifft, die auf OTHERWISE folgende Anweisung abgearbeitet. Nach deren Abarbeiten der durch CASE bzw. OTHERWISE ausgewählten Anweisungen erfolgt die Programmfortsetzung mit der auf ENDCASE folgenden Anweisung.
Innerhalb einer Befehlsdatei dürfen CASE-Anweisungen nicht ineinander geschachtelt werden (im Gegensatz zur IF-Anweisung)!

6.3. Wiederholung

Mittels der WHILE-Anweisung wird eine Anweisung (Anweisungsfolge) abgearbeitet, solange die angeführte Bedingung erfüllt ist. Dabei wird *zuerst* geprüft, ob die Bedingung zutrifft, das heißt, gegebenenfalls wird der Block keinmal ausgeführt (Bild 4).



```
Beispiel:
...
? 'Erfaßte Personen aus der Stadt Dresden'
DO WHILE NOT EOF
   DISPLAY NAME, VORNAME FOR
   ORT = 'Dresden'
   SKIP
ENDDO
```

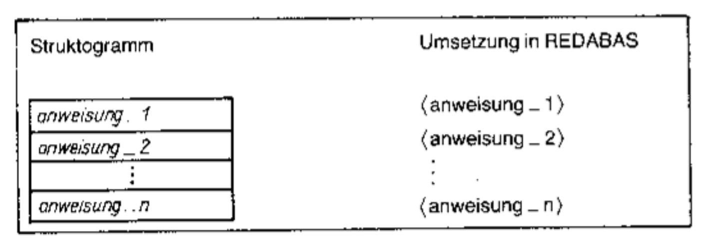
Die aufgeführte Anweisung (Anweisungsfolge) wird immer nur dann ausgeführt, wenn die Bedingung wahr ist; nach ihrer Ausführung wird die Bedingung erneut ausgewertet. DO WHILE-Verschachtelungen sind zulässig.

6.4. Bemerkungen

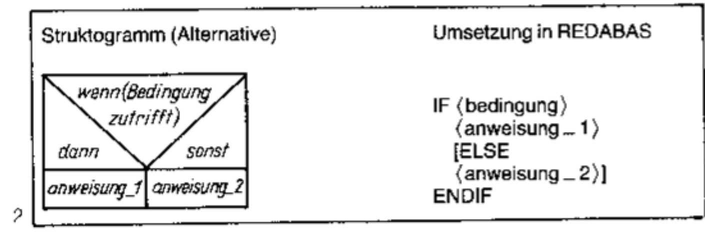
In REDABAS sind zwei wichtige Programmstrukturen *nicht* enthalten: Es fehlen die bedingte Wiederholung mit Endabfrage (Abfragebedingung am Ende), meist durch REPEAT ... UNTIL realisert, sowie die Zählschleife (n-maliges Durchlaufen einer Schleife entsprechend einem gegebenen Wert n), meist durch FOR ... TO dargestellt. Beide Strukturen müssen durch die in REDABAS vorhandenen Programmelemente ersetzt werden.

Im Sinne der strukturierten Programmierung verzichtet REDABAS auf die in vielen anderen Programmiersprachen vorhandene Sprunganweisung GOTO.

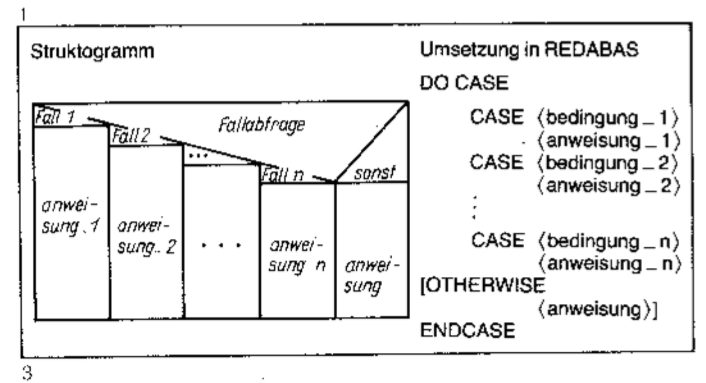
Beim Umgang mit den unter Pkt. 6.2. und 6.3. behandelten Anweisungen ist insbesondere wichtig, die richtige Stellung der jeweiligen END-Anweisungen (ENDDO, ENDIF, END-



1



2



3

Bild 1 Sequenz von Strukturblöcken
Bild 2 Alternative
Bild 3 Fallunterscheidung
Bild 4 Abweisschleife/Kopfgesteuerte Schleife

Struktogramm	Umsetzung in REDABAS
wiederhole (solange Bedingung zutrifft) / anweisung	DO WHILE ⟨bedingung⟩ ⟨anweisung⟩ ENDDO

4

CASE) zu analysieren; hier kommt bei falscher Stellung leicht die gesamte Programmlogik durcheinander!

7. Weiteres Arbeiten mit Speichervariablen

Die Verwaltung von Speichervariablen stellt insbesondere bei Befehlsdateien eine wichtige Anwendung in REDABAS dar. Da sie temporär im Arbeitsspeicher verwaltet werden, ist es aufgrund der Beschränkung auf eine maximale Zahl von 64 notwendig, beim Programmtest einen entsprechenden Überblick zu behalten:

LIST MEMORY

listet alle temporären Speichervariablen mit deren Inhalt, Größe und Typ auf.
Solche Speichervariablen bleiben solange definiert, bis REDABAS mit der QUIT-Anweisung verlassen wird (sofern sie nicht gezielt gerettet werden).
Außer den bereits behandelten Anweisungen zur Arbeit mit Speichervariablen sollen folgende noch kurz erläutert werden:

STORE ⟨ausdruck⟩ TO ⟨speichervariable⟩

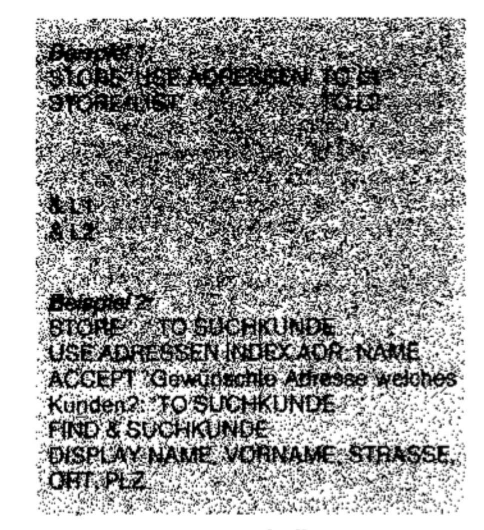
Mittels dieser Anweisung kann einer Speichervariablen der Wert eines Ausdrucks beliebigen Datentyps zugewiesen werden.
Existiert die Speichervariable noch nicht, wird sie mit dieser Anweisung angelegt. Die Zuweisung erfolgt unabhängig davon, ob die Speichervariable eventuell vorher mit einem anderen Datentyp besetzt war.

Beispiele:
```
STORE 'Montag' TO TAG
STORE 5 × 4 - 3 TO ERGEBNIS
```

Aufgrund der zahlenmäßigen Begrenzung der Speichervariablen sind weitere Anweisungen nützlich wie:
– RELEASE bzw. CLEAR
zum vollständigen/teilweisen Löschen temporärer Speichervariablen
– SAVE
zum Sichern temporärer Speichervariablen auf Diskette, die mit
– RESTORE
wieder in den Arbeitsspeicher geladen werden können.

Makroanweisung & ⟨speichervariable⟩

Die Makroanweisung in REDABAS bietet Möglichkeiten, wiederholte Eingaben z. B. von Anweisungszeilen zu vermeiden bzw. zu verkürzen sowie Anweisungen, bei denen z. B. nur Konstanten für den Suchbegriff zulässig sind, variabel zu gestalten. Sie dient damit der indirekten Adressierung von Anweisungsobjekten wie Feldern, Datensätzen, Dateien oder Anweisungen über Speichervariablen.
Trifft REDABAS in Anweisungen oder bei der Auswertung von Ausdrücken auf das Zeichen „&" (Ampersand), wird zum Zeitpunkt der Befehlsabarbeitung der in der Speichervariablen befindliche Inhalt ausgewertet. Das bedeutet, daß zur Speichervariablen zuvor eine Eingabe bzw. Zuweisung erfolgt sein muß.

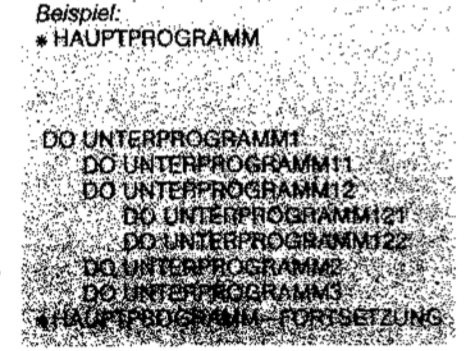

Beispiel 1:
```
STORE 'USE ADRESSEN' TO L1
STORE 'LIST'         TO L2
&L1
&L2
```
Beispiel 2:
```
STORE ' ' TO SUCHKUNDE
USE ADRESSEN INDEX ADR. NAME
ACCEPT 'Gewünschte Adresse welches
Kunden?: 'TO SUCHKUNDE
FIND & SUCHKUNDE
DISPLAY NAME, VORNAME, STRASSE,
ORT, PLZ
```

8. Unterprogrammtechnik

Unterprogramme sind in REDABAS Teilprogramme mit gleichem Aufbau wie Befehlsdateien sowie thematisch begrenzten Anweisungsfolgen. Schachtelungen sind möglich. Der *Aufruf* von Unterprogrammen erfolgt analog dem Aufruf von Befehlsdateien:
aus der Anweisungsebene bzw. übergeordneten Befehlsdatei heraus. Eine Parameterübergabe (Parameterliste) beim Aufruf ist *nicht* zulässig. Die *Rückkehr* in die aufrufende Ebene erfolgt mit

RETURN

Dabei wird die Abarbeitung des Unterprogramms beendet; die Programmsteuerung geht an die aufrufende Ebene zurück.


Beispiel:
```
* HAUPTPROGRAMM

DO UNTERPROGRAMM1
   DO UNTERPROGRAMM11
   DO UNTERPROGRAMM12
      DO UNTERPROGRAMM121
      DO UNTERPROGRAMM122
   DO UNTERPROGRAMM2
   DO UNTERPROGRAMM3
* HAUPTPROGRAMM – FORTSETZUNG
```

9. Hinweise zur Erstellung und zum Test von Befehlsdateien [2–4]

Die Analyse eines Anwendungsproblems hat in REDABAS eine ebenso große Bedeutung wie in anderen Programmiersprachen oder bei der Nutzung anderer Standardsoftware. Mit dem Vorteil von REDABAS, bei der Analyse/Programmierung nicht berücksichtigte Aufgaben diese später im direkten Dialog zu lösen, sollte sparsam umgegangen werden. Bediener ohne Vorkenntnisse kommen sonst kaum in die Lage, solcherart Programme effektiv zu nutzen. Damit steigen die Anforderungen an Programmsicherheit, Programmkomfort und verständlichen Programmablauf. Wichtige Gesichtspunkte bei der *Erstellung von Befehlsdateien* sind damit u. a. (siehe */2/*, */3/*):

– Modulare, übersichtliche Programmierung zur Erstellung der einzelnen Arbeitsabläufe
– Prüfung des Datenvolumens
– Prüfung der Art der anzulegenden Dateien (z. B. für Stamm-, Bewegungsdaten)
– Nutzung von Bildschirmmasken
– Paßwortschutz
– optimale Gestaltung von Sortierkriterien
– Datensicherungsmaßnahmen.

Zur *Steigerung der Interpreter-Leistung* von REDABAS sind folgende allgemeine Hinweise wichtig:
Insbesondere bei mittelgroßen und großen Datenbeständen macht sich die interpretative Abarbeitung laufzeitmäßig bemerkbar.

Hier ist
1. genau der Rahmen der anzustrebenden Problemlösung abzustecken (z. B. nicht nach allen möglichen Variablen zu indizieren, die aber nur teilweise benötigt werden)
2. Programmierung aus dem Stegreif zu vermeiden,
3. nach Erstellung einer ersten Programmvariante nach weiteren, effektiveren zu suchen
4. überhaupt zu trennen, was mit REDABAS im Dialog und welche Aufgabe mittels Programmierung zu lösen ist. (Der Einsatz beider Möglichkeiten führt in den meisten Fällen zu einem optimalen Ergebnis.)

Für Programmtestläufe sind insbesondere SET-Anweisungen nützlich, wie

SET ECHO ON

zum Anzeigen aller ausgeführten Anweisungen einer Befehlsdatei während der Laufzeit auf Bildschirm (Fehlersuche!),

SET STEP ON

zur Einzelschrittabarbeitung einer Befehlsdatei mit nachfolgender Abfrage zur weiteren Abarbeitung,

SET TALK ON

zur Anzeige von REDABAS-Systemmeldungen (Voreinstellung),

SET DEBUG ON

zur Ausgabe der mit SET ECHO ON/SET STEP ON erzeugten zusätzlichen Ausgaben auf Drucker.

Einzelne Möglichkeiten zur Erhöhung der Laufzeiteffizienz sind zum Beispiel:
● bei REDABAS-Schlüsselwörtern nur die ersten 4 Zeichen zu verwenden (reicht dem Interpreter zur Identifizierung):

aus	wird
CREATE	CREA
DISPLAY	DISP

● die Verwendung der LOOP-Anweisung in DO WHILE-Schleifen zur sofortigen Rückkehr bei Abbruchbedingungen an den Schleifenanfang, ohne die Anweisungen bis zum nächsten ENDDO erst auszuführen (Zeitersparnis)
● statt WAIT die ACCEPT-Anweisung einzugeben (Bei WAIT erfolgt Nachladen von Diskette)

wird fortgesetzt

Neues vom PC

Kürzlich stellte IBM ihr Personal System/2 vor. Neu sind nicht nur Architektur und Technologie - es kommen VLSI-Bauelemente (beim Modell /80 wird z. B. der 1-MBit-Chip verwendet) und die SMT (Surface Mount Technology) zum Einsatz - sondern auch ein Großteil der Peripherie.
Mit dieser Computersystemfamilie will die IBM ihre Monopolstellung weiter festigen bzw. ausbauen. Das macht auch der Einsatz des neuen Betriebssystems (BS) Operating System /2 (OS/2) deutlich. Allerdings hält sich der Konzern mit dem DOS 3.3 - alle Modelle laufen auch unter diesem BS - die „Hintertür" zur PC-„Welt" offen.
Nachfolgend sollen die neuen Systeme kurz vorgestellt werden.

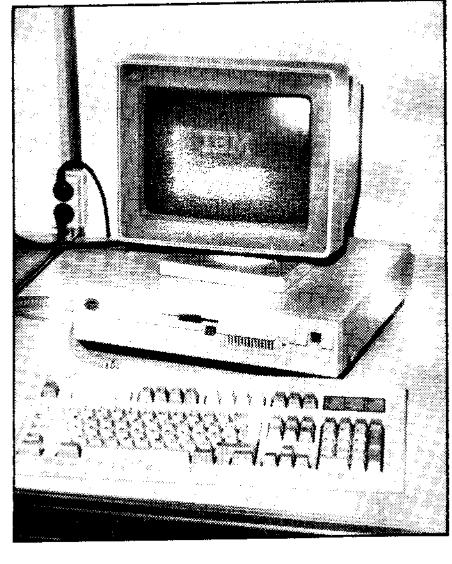
Die Architektur des Personal Systems /2 (PS/2) erlaubt einerseits einen sehr hohen Datentransfer innerhalb des Systems, andererseits garantiert sie eine gewisse Kompatibilität zu den Großrechnern ebenso wie zu den PC des Herstellers.
Zur PS/2-Familie gehören vier Modellgruppen. Dabei sind insgesamt acht Modelle möglich. Ebenfalls neu sind vier Bildschirmgeräte, vier Drucker und die optische Platteneinheit 3363.
Je nach Modell werden die folgenden Prozessortypen von Intel verwendet: 8086 (16-Bit-μP mit einer Taktfrequenz von 8 MHz), 80286 (16/24-Bit-μP mit 10 MHz), 80386 (32-Bit-μP mit 16 bzw. 20 MHz).

Modell 30

Es ist das kleinste der neuen Reihe und als Teilmodell ausgeführt. Wichtige Parameter sind: Prozessor 8086, 640 KByte Hauptspeichergröße, Sockel für mathematische Coprozessoren, je eine serielle und parallele Schnittstelle, Anschluß für eine Maus sowie ein integrierter Farbgrafikadapter MCGA (Multi Colour Graphics Array). Wahlweise sind zwei 3,5-Zoll-Diskettenlaufwerke mit je 720 KByte Speicherkapazität oder ein Diskettenlaufwerk und eine 20-MByte-Festplatte möglich. Als BS findet DOS 3.3 Verwendung.

Modell 50

Dieses System, es ist ebenfalls als Tischgerät ausgeführt, hat einen 1 MByte-Hauptspeicher, ein 3,5-Zoll-Diskettenlaufwerk mit 1,44 MByte Kapazität, eine 20-MByte-Festplatte, drei 16-Bit-Erweiterungssteckplätze, einen Sockel für den mathematischen Coprozessor und die üblichen Schnittstellen.
Für eine grafische Verarbeitung sind durch das Video-Graphics-Array(VGA)-Verfahren 640 mal 480 Punkte direkt adressierbar, wobei von 262 000 möglichen Farben 256 gleichzeitig auf dem Bildschirm dargestellt werden können. Als Betriebssysteme können DOS 3.3 oder OS/2 verwendet werden. Der Hauptspeicher des mit einem Mikroprozessor 80286 bestückten Computers läßt sich auf maximal 7 MByte erweitern.



Unser Bild zeigt das kleinste Modell des PS/2 auf der diesjährigen Internationalen Messe Poznan
Foto: Weiß (1)

Modell 60

Dieses Modell ist in den wesentlichsten Parametern mit dem 50er identisch. Die Systemeinheit ist jedoch als Standmodell ausgeführt und besitzt eine 44-MByte- bzw. 70-MByte-Festplatte.

Modell 80

Leistungsstärkstes Computersystem dieser Familie ist das Modell 80, das es in drei Ausführungen gibt. Je nach Ausführung arbeitet der 32-Bit-Mikroprozessor 80 386 im 16- oder 20-MHz-Takt. Die Hauptspeicherkapazitäten betragen 1 bzw. 2 MByte. Zur Konfiguration gehören ein 1,44-MByte-Diskettenlaufwerk und – je nach Ausführung – 44, 70 oder 115 MByte an Plattenspeichern. Neben vier 16-Bit- und drei 32-Bit-Erweiterungssteckplätzen sind die gleichen Erweiterungen möglich wie beim Modell 60 bzw. 50. Auch das 80er Modell läuft unter beiden BS.

Peripherie

Neben den eigentlichen Systemeinheiten wurden neue Bildschirme und Drucker vorgestellt. So ein 30,5-cm-Monochrombildschirm (8503), der schwarze Schrift auf weißem Grund oder umgekehrt bietet und die Darstellung von 64 verschiedenen Grauwertstufen erlaubt.
Der leistungsfähigste der neuen Farbbildschirme (8514) ermöglicht die Adressierung von bis zu 1024 × 768 Pixel bei 92 Pixel je Zoll.
Bei den Druckern handelt es sich um den Grafikdrucker 4201-002 mit drei integrierten Schriftarten, zwei Qualitätsstufen (Schnelldruck und Textqualität) und einer Geschwindigkeit von bis zu 240 Zeichen/s sowie zwei Matrixdrucker 4207 und 4208 mit 24-Nadel-Kopf- und einen Thermodrucker (5202).
Von besonderem Interesse ist die optische Platteneinheit (3363). Dabei wird eine Technologie verwendet, bei der die Daten mit Laserstrahlen auf das Speichermedium Kunststoffscheibe geschrieben werden. Eine Platte – Durchmesser rund 13 cm – kann bis zu 200 MByte speichern. Mit einer Anschlußkarte können zwei dieser Einheiten an das PS/2 Modell 30 bzw. an IBM PCs angeschlossen werden. Bis zu acht Platteneinheiten können mit den Modellen 50, 60 und 80 verbunden werden.
Die optische Platte ist ein sogenannter WORM-Speicher (Write-once-read-many). Sie eignet sich insbesondere zur Speicherung und Archivierung umfangreicher Datenbestände.

Software

Wie schon erwähnt, verwendet IBM das bekannte DOS 3.3, um die Kompatibilität zwischen PC-Anwendungen und solchen des PS/2 herzustellen. Dabei enthält das DOS 3.3 eine Reihe neuer Funktionen und Befehle und bietet in Verbindung mit Top View 1.12 Multitasking-Unterstützung bei einer Reihe von Anwendungen. Das neue Betriebssystem OS/2 soll die Vorteile der neuen Prozessoren, der großen Speicher, der 32-Bit-Micro-Channel-Architektur für den Benutzer besser als das DOS verfügbar machen, und es erlaubt uneingeschränkt Multitasking. Vom OS/2 werden die Programmiersprachen BASIC Compiler /2, Macro-Assembler /2, FORTRAN /2, COBOL /2 und Pascal Compiler /2 unterstützt.

I. P.

Tastaturabfrage beim KC 85/3

Beim BASIC-Interpreter des KC 85/3, als auch des KC 85/2, stehen zwei Anweisungen zur Tastatureingabe zur Verfügung. Die INPUT-Anweisung erfordert als Abschluß stets das Betätigen der ENTER-Taste. Soll über die Tastatur ein Programmablauf gesteuert werden, z. B. eine Menüauswahl, ist es oftmals günstig, auf einen Tastendruck sofort zu reagieren. Dafür steht die INKEY-Anweisung zur Verfügung. Wird diese ständig aufgerufen, so ist sie relativ langsam und bedingt viele auf dem Bildschirm sichtbare Bildwiederholspeicherzugriffe. Eine Verbesserung bringt hier der direkte Zugriff auf die Speicherzelle, in welcher vom Betriebssystem der Tastenkode bereitgestellt wird. Dies ist die Adresse 1FDH. Die Abfrage kann mit PEEK(509) erfolgen. Ein Warten auf die ENTER-Taste ist z. B. durch diese Zeile möglich:

```
100 IF PEEK(509)=13 THEN 200:ELSE 100
```

Ist keine Taste betätigt, ist der Kode 0. Der Tastenkode steht so lange bereit, wie die Taste gedrückt bleibt, auch bei mehrmaliger Abfrage. Das hier beschriebene Verfahren unterliegt, im Gegensatz zur INKEY-Anweisung, nicht der Tastaturfolgefrequenz des Autorepeat.

K.-D. Kirves

Der Modul M026 FORTH für die Kleincomputer KC 85/2 und KC 85/3

Dr. Werner Domschke
VEB Mikroelektronik „Wilhelm Pieck“
Mühlhausen
Klaus Katzmann
VEB Uhrenwerke Ruhla

Der Vorzug des Kleincomputersystems KC 85 des VEB Mikroelektronik „Wilhelm Pieck“ Mühlhausen besteht u. a. in seiner Modularität, d. h. das System läßt sich vom Anwender entsprechend seinen Anforderungen und Wünschen aufrüsten. Das erfolgt durch Module, die den Speicher des KC 85 erweitern können (RAM oder ROM) bzw. die Schnittstellen zur Umgebung schaffen.
Neben dem bereits vorhandenen Modulsortiment stellt der Modul M026 FORTH einen weiteren Beitrag zum Systemausbau dar.
Der Modul basiert auf der Hardware des Moduls M025 USER PROM 8K und ist neben den Modulen M012 TEXOR und M027 DEVELOPMENT eine Modfikation desselben.
Mit dem Modul M026 wird der Speicher des KC 85 um 8KByte ROM erweitert. Dem Anwender steht damit neben BASIC die höhere Programmiersprache FORTH zur Verfügung /1/.

1. Beschreibung von FORTH

FORTH ist eine Sprache, die 1969 für die 4. Rechnergeneration entworfen wurde. Sie besitzt eine Reihe von Eigenschaften, die sie von den geläufigen Programmiersprachen wesentlich unterscheidet.
FORTH hat viele Vorteile, die eine Implementierung auf dem KC 85/2 bzw. KC 85/3 rechtfertigen. Die Mehrzahl der Umsteiger von BASIC auf FORTH möchten für „ernsthafte“ Programme keine andere Sprache mehr verwenden.

1.1. Wesentliche Eigenschaften von FORTH

FORTH gehört zu den verknüpften interpretativen Sprachen (TIL - threaded interpretive language). Dadurch besitzt FORTH eine Reihe von nützlichen Eigenschaften, die sie fast zur idealen Mikrorechnersprache macht /2/.

● FORTH besitzt die Möglichkeit, die Elemente von FORTH interpretativ abarbeiten zu können, wie das i. allg. mit BASIC-Programmen erfolgt.
In FORTH existieren mehr sofort ausführbare Befehle als z. B. in BASIC.
Diese interpretative Abarbeitung eingegebener Befehle ist ein wichtiger Vorteil bei der Programmerarbeitung. Fehler können sofort gefunden und beseitigt werden.

● FORTH besitzt die Möglichkeit, getestete Programme oder Programmteile in die interne Darstellung zu übersetzen, also zu compilieren. Es werden damit neue direkte ausführbare Befehle erzeugt, die den vorhandenen Interpreter erweitern. Elemente der neuen Befehle sind die schon vorhandenen Befehle des Interpreters. Die neu erzeugten Befehle, die in FORTH als Worte bezeichnet werden, können selbst wieder Bausteine neuer Worte sein.
Damit erreicht man modular aufgebaute Programme.

● FORTH besitzt die direkte Testmöglichkeit einzelner Befehle oder Befehlsgruppen. Man muß nicht wie beispielsweise in PASCAL zum Testen einzelner Unterprogramme (Subroutinen) extra ein Hauptprogramm schreiben. Da die verwendeten FORTH-Worte vorher bereits auf Fehlerfreiheit getestet wurden, ergeben sich in FORTH relativ zuverlässige Programme.

● In FORTH müssen bereits compilierte Worte bei Einbinden in neue Programme (neue FORTH-Worte) nicht als Quellcode vorliegen oder nicht mit einem Linker eingebunden werden. Damit wird Speicherplatz für die Quelltextdatei und Übersetzungszeit gespart.

● Der FORTH-Interpreter benötigt relativ wenig Speicherplatz. Das KC-FORTH besitzt eine Länge von 8 KByte bei einem relativ hohen Programmierkomfort.

● FORTH erzeugt einen sehr effizienten Programmcode. Gegenüber BASIC sind die Programme 2- bis 10mal kürzer, je nach Anwendungsbereich. Selbst gegenüber Assemblerprogrammen kann FORTH gleichlange oder u. U. kürzere Programme liefern.

● FORTH ist wegen seiner Struktur relativ schnell in der Abarbeitung. Je nach Problemstellung sind FORTH-Programme 1.5- bis 15mal so schnell wie BASIC-Programme und nur 2- bis 5mal langsamer als Maschinenprogramme. Zeitkritische Probleme lassen sich in Maschinencode formulieren.

● FORTH ist maschinennah.
Der Anwender kann die Ressourcen des Computers voll nutzen, da elementare und komfortable Speicher- und Ein-/Ausgabeoperationen als FORTH-Worte direkt vorhanden sind. Zeitkritische Aktionen lassen sich im Maschinencode programmieren, ohne daß später erkennbar ist, welches FORTH-Wort selbst in FORTH oder in Maschinensprache erarbeitet wurde.
Da FORTH eindeutig definierte Schnittstellen zur Parameterübergabe zwischen den FORTH-Worten besitzt, ist ein Einpassen von Maschinenprogrammen in das FORTH-Konzept relativ einfach.

● FORTH fordert die strukturierte Programmierung.
FORTH-Worte besitzen nur einen „Eingang“ und nur einen „Ausgang“. Für unbedingte oder bedingte Sprünge („GOTO“) existieren in FORTH keine vom Anwender nutzbaren Konstruktionen, die auch nicht benötigt werden.
Dies erzwingt bei der Programmerstellung eine hohe Disziplin. Damit steigt die Übersichtlichkeit und die Zuverlässigkeit von Programmen.

● FORTH ist ein komplettes Programmentwicklungssystem. Dem Anwender stehen alle Hilfsmittel zur Verfügung, die für eine Programmentwicklung erforderlich sind (z. B. Editor, Interpreter, Compiler).

● Mit FORTH können alle Ressourcen der Hardware vollständig genutzt werden. Die Schnittstellen zur Peripherie sind sehr einfach gehalten. Da FORTH durch seine Erweiterbarkeit sehr flexibel ist, ist es auch an veränderte Hardwarebedingungen oder Peripherie problemlos anpaßbar.

● FORTH ist in hohem Maße portabel. Da der FORTH-Interpreter eine nahezu hardwareunabhängige Programmieroberfläche schafft, sind Programme des KC-FORTH auch ohne wesentliche Änderungen auf 16-Bit-Maschinen lauffähig. Voraussetzung dabei ist aber die Verwendung des FORTH-Standardwortschatzes, also dürfen z. B. die Farb- und Grafikbefehle nicht in dem umzusetzenden Programm enthalten sein.

Die genannten Vorteile bringen aber auch einige Ungewöhnlichkeiten mit sich. Der Programmierer muß sich an den anderen Programmierstil gewöhnen. Die größten Probleme für „Umsteiger“, z. B. von BASIC auf FORTH, bereitet die Umgekehrte Polnische Notation (UPN) beim Aufruf der einzelnen FORTH-Worte. Aber gerade die UPN macht die Parameterübergabe von einem FORTH-Wort zu dem nächsten so einfach und überschaubar.
Als Wertespeicher gibt es in FORTH den Stack („Zahlenstapel“), über den die Parameterübergabe erfolgt. Jedes FORTH-Wort entnimmt vom Stack die Anzahl der erforderlichen Argumente und hinterläßt auf dem Stack das Ergebnis.
Der Stack ist ein LIFO-Speicher (last in, first out), d. h., jede neue Zahl wird auf den Stapel „obenauf“ gelegt und von oben wieder entfernt.
Anhand einiger FORTH-Worte soll die Vorgehensweise verdeutlicht werden.

Verwendet werden folgende FORTH-Wörter:

+ – erwartet auf dem Stack zwei Zahlen, addiert sie und legt das Ergebnis wieder auf den Stack ab.

∗ – erwartet auf dem Stack zwei Zahlen, multipliziert sie und legt das Ergebnis wieder auf den Stack ab.

. – erwartet auf dem Stack eine Zahl, konvertiert sie und gibt diese auf dem Bildschirm aus.

Beispiel 1: 45 + 234 = 279

Eingabe		45	234	+	.
Ausgabe					279
Stack	–	45	234	279	–
		–	45	–	
			–		

Beispiel 2: (7 + 5) ∗ 8 = 96

Eingabe		7	5	+	8	∗	.
Ausgabe							96
Stack	–	7	5	12	8	96	–
		–	7	–	12	–	
			–		–		

Das letzte Beispiel verdeutlicht auch, daß man in FORTH keine Klammern bei der Be-

rechnung komplexer mathematischer Ausdrücke benötigt.

1.2. Aufbau und Arbeitsweise von FORTH

FORTH besitzt einen Satz von direkt ausführbaren Worten, die in Maschinensprache geschrieben sind. Sie werden als Primitiv-Worte bezeichnet. Diese Worte bilden den Kern des FORTH-Interpreters. Aus diesen Worten können andere FORTH-Worte gebildet werden. Die Worte, die selbst aus FORTH-Worten bestehen, werden Sekundär-Worte genannt.
Die einzelnen FORTH-Worte sind im Speicher in einer Kette angeordnet. Diese Kette wird Wörterbuch (dictionary) genannt. Jedes FORTH-Wort besitzt einen Zeiger auf das nächste FORTH-Wort. Die Wörter sind damit verkettet. Somit läßt sich dieses Wörterbuch recht einfach nach einem bestimmten Wort durchmustern.
Jedes FORTH-Wort besteht aus dem Kopf, in dem der Name des Wortes als ASCII-Zeichen, die Länge des Namens sowie die Adresse des nächsten Wortes steht, und dem Körper, in dem der Typ des Wortes (Primitiv-, Sekundärwort, Variable, Konstante usw.) und bei Primitiv-Worten der ausführbare Maschinencode, bei Sekundärworten die Liste der Adressen der aufzurufenden FORTH-Worte, bei Variablen und Konstanten der Wert usw. steht.
Ein spezieller Programmteil organisiert bei Sekundärworten den Aufruf der im Körper stehenden FORTH-Worte. Dieser Teil wird innerer Interpreter genannt. Im Gegensatz dazu organisiert der äußere Interpreter die Suche des gerade eingegebenen Wortes im Wörterbuch und veranlaßt die Ausführung.
Die interpretative Abarbeitung eines FORTH-Wortes geschieht in der folgenden Weise:

– Analyse des vom Bediener eingegebenen Textes auf einzelne Worte (Leerzeichen sind Trennzeichen).
– Suche des ersten eingegebenen Wortes im Wörterbuch des Interpreters, beginnend mit dem letzten compilierten FORTH-Wort.
– Wurde eine Übereinstimmung zwischen dem eingegebenen Wort und dem Namen eines FORTH-Wortes gefunden, wird die Adresse dem inneren Interpreter übergeben und das FORTH-Wort ausgeführt.
– Wurde kein FORTH-Wort gefunden, das mit dem eingegebenen Wort übereinstimmt, wird versucht, das Wort als Zahl zu interpretieren. Diese Zahl wird in der internen Darstellung auf dem Stack abgelegt.
– Läßt sich das eingegebene Wort auch als Zahl nicht interpretieren, wird eine Fehlermeldung ausgegeben.

Einige FORTH-Worte schalten vom interpretativen auf den compilierenden Betrieb um (z. B. der Doppelpunkt :) bzw. zurück (z. B. das Semikolon ;). Neue FORTH-Worte werden so gebildet, daß die Kette der FORTH-Worte verlängert, das Wörterbuch also erweitert wird. Am Ende des Wörterbuches wird ein Kopf mit dem neuen Namen des zu compilierenden FORTH-Wortes angelegt, der Zeiger auf das vorher definierte FORTH-Wort gestellt und der Körper entsprechend eingerichtet. Programme in FORTH bestehen prinzipiell aus einem speziell erweiterten und angepaßten FORTH-Interpreter.

An folgendem Beispiel sei das dargestellt /3/. Eine Haushaltswaschmaschine wird durch einen Computer gesteuert, der in FORTH programmiert wird. Folgende FORTH-Worte sind vom Hersteller vorgegeben:

FUELLEN – die Maschine wird mit Wasser gefüllt
LEEREN – die Maschine wird leergepumpt
XX GRAD – einstellen des Temperaturschalters auf xx Grad
HEIZEN – die Heizung wird bis zum Erreichen des am Temperaturschalters mit xx GRAD eingestellten Wertes eingeschaltet
WASCHEN – die Trommel wird eingeschaltet
YY MINUTEN – einstellen des Zeitschalters auf yy Minuten
Dieser schaltet nach Ablauf der Zeit die Trommel aus.

Mit diesem „Grundwortschatz“ kann die Hausfrau ihre speziellen Waschprogramme selbst zusammenstellen.
Im einfachsten Fall geschieht das durch Aufruf der einzelnen Worte in der entsprechenden Reihenfolge. Man kann den Interpreter aber auch nach den Bedürfnissen der Hausfrau erweitern, indem neue Worte compiliert werden.
Das Spülprogramm könnte so definiert werden:

: SPUELEN FUELLEN 5 MINUTEN WASCHEN LEEREN;

Mit dem Doppelpunkt wird auf compilierenden Betrieb umgeschaltet und das FORTH-Wörterbuch um das FORTH-Wort SPUELEN erweitert. Das Semikolon beendet das Wort und schaltet auf interpretativen Betrieb um. Dieses Wort kann jetzt direkt aufgerufen werden, dann spült die Maschine 5 Minuten lang die Wäsche, oder es kann in ein anderes noch komplexeres FORTH-Wort eingebaut werden.

Beispiel:

:HAUPTWAESCHE FUELLEN HEIZEN 10 MINUTEN WASCHEN LEEREN SPUELEN SPUELEN ;

Die Temperatureinstellung GRAD erfolgt hier nicht im Wort HAUPTWAESCHE. Die Temperatur muß deshalb vor dem Aufruf von HAUPTWAESCHE eingestellt werden.

: WOLLE 30 GRAD HAUPTWAESCHE;
: SYNTHETIC 40 GRAD HAUPTWAESCHE;
: WEISS 60 GRAD HAUPTWAESCHE;

Dies sind einige Definitionen, die zum „normalen Programm“ einer Waschmaschine gehören. Dies läßt sich aber von der Hausfrau beliebig variieren.

: VORWAESCHE FUELLEN 40 GRAD HEIZEN 15 MINUTEN WASCHEN LEEREN;
: KOCHEN VORWAESCHE 95 GRAD HAUPTWAESCHE;

Dieses Waschmaschinenbeispiel spiegelt ein wesentliches Anwendungsgebiet für FORTH wider: die Regelungs- und Steuerungstechnik. Aber auch in anderen Anwendungsbereichen können die Vorzüge von FORTH andere Programmiersprachen teilweise oder ganz ersetzen.

Da FORTH allerdings schwerer zu erlernen ist als z. B. BASIC oder PASCAL, wird FORTH nicht die Verbreitung finden wie diese Sprachen. FORTH ist eine günstige Alternative zur Assemblerprogrammierung.

2. KC-FORTH

Das im Modul M026 implementierte FORTH ist zurückzuführen auf die von der FORTH Interest Group für den Prozessor Z80 veröffentlichte Version (fig-FORTH). Diese wird in /4/ sehr ausführlich beschrieben. Das fig-FORTH basiert auf einer Hardware mit mindestens einem Floppy-Disk-Laufwerk als Massenspeicher. Da dies beim KC85 des VEB Mikrolektronik „Wilhelm Pieck“ Mühlhausen nicht gegeben ist, wurden einige Veränderungen vorgenommen, die die Besonderheiten des KC85 berücksichtigen und dem Programmierer zugänglich machen.

2.1. Sprachumfang von KC-FORTH

Der Umfang von KC-FORTH entspricht im wesentlichen dem in /4/ erläuterten fig-Standard. Abstriche wurden bei Doppeldefinitionen (THEN, END) und bei der Massenspeicherarbeit (+BUF, BUFFER, UPDATE, FLUSH) gemacht. Für den KC nicht benötigte Worte wurden nicht implementiert, andere Worte wurden modifiziert (FIRST, LIMIT, BLOCK, OFFSET) oder erhielten andere Bedeutung. Damit wurde der Magnetbandarbeit als Massenspeicher entsprochen.

Zusätzlich wurde ein Editor aufgenommen, dessen Vorbild auf der Zeilenebene auf den in /4/ angegebenen fig-Vorschlag zurückgeht. Weiterhin enthält das KC-FORTH eine Reihe von Worten, die es dem Anwender gestatten, alle Fähigkeiten des KC85 in einfacher Weise zu nutzen (Grafik, Tonausgabe, Farbe, Modulsteuerung, Bildschirmsteuerung u. a.). Die in /5/ angegebene Liste der häufigsten FORTH-Worte ist mit den genannten Ausnahmen auch auf das KC-FORTH zutreffend.

Justieren des Tonkopfes

Der Tonkopf des Kassettenmagnetbandgerätes (KMBG) sollte von Zeit zu Zeit justiert werden. Dazu wurde auf den Programmkassetten R 0191, R 0192 und R 0193 vom VEB Robotron-Vertrieb Berlin ein Prüfsignal aufgezeichnet und auf die Kassette im Rahmen des Informationsprogrammes eine einfache Justiervorschrift eingegeben. Schlecht justierte Tonköpfe können das Einlesen von Programmkassetten unmöglich machen. Dieser Effekt tritt besonders dann in Erscheinung, wenn auf einem fremden KMBG aufgezeichnete Programme mit dem eigenen KMBG eingelesen werden sollen und eines dieser beiden Geräte einen dejustierten Tonkopf besitzt.

Dr. G. Kleinmichel

2.2. Magnetbandarbeit

Wesentliches Merkmal der KC-FORTH-Massenspeicherarbeit ist eine im RAM eingerichtete virtuelle Diskette (RAM-FLOPPY).
Hierdurch ist eine weitgehende Kompatibilität mit der diskettenorientierten fig-FORTH-Version gegeben. Der Anwender kann auch im KC-FORTH Programme (Quellcode) oder Daten in Screens speichern und Screens als neue Eingabequelle definieren.
Als Screen wird eine Informationsmenge verstanden, die sich wegen ihrer Größe günstig auf dem Bildschirm darstellen läßt. Ein Screen hat beim KC-FORTH eine Größe von 16 Zeilen zu 32 Zeichen (512 Bytes Information und zwei Bytes Blocktrennung). Die Anzahl der Screens ist von der Speicherausrüstung des KC abhängig. Der Anwender hat die Möglichkeit, die Lage und Größe der virtuellen Diskette im Speicher mit den beiden User-Variablen FIRST und LIMIT selbst festzulegen. In der Grundinitialisierung stehen 8 Screens zur Verfügung. Der Anwender kann in der Grundausrüstung bei einer Festlegung von FIRST auf IFE0H und LIMIT auf 4000H den Bereich auf 16 Screens ausdehnen. Werden Speichererweiterungsmodule verwendet, so lassen sich mit dem M011 64KByte RAM ohne Speicherumschaltung 79 Screens und mit Speicherumschaltung (SWITCH) 142 Screens nutzen. Die Verwendung weiterer RAM-Module zur Erweiterung der virtuellen Diskette ist problemlos möglich.
Die Speicherung der Screens auf Magnetband übernimmt das Wort CSAVE in der folgenden Art:

nl n2 CSAVE zzzzz

Speichern der Screens nl bis n2 mit dem Namen zzzzz. Als Dateikennzeichen wird „(F)" eingetragen.
Dateien können mit CLOAD in die virtuelle Diskette geladen werden.

nl n2 CLOAD

Die Screens nl bis maximal n2 werden mit den Daten der nächsten gefundenen Datei gefüllt. Eine Angabe des Namens ist nicht erforderlich. Ist die Datei zu lang, wird nach dem Screen n2 abgebrochen.
Die Anzahl der Screens, die vom Anfang der Datei her ignoriert werden sollen, wird in der User-Variablen OFFSET festgelegt.
Auf der untersten Ebene der Magnetbandarbeit steht das Wort R/W, mit dem 128-Byte-Blöcke vom Speicher auf Magnetband geschrieben bzw. vom Magnetband in den Speicher gelesen werden können.

2.3. Der KC-FORTH-Editor

Um Texte in Screens zu erstellen und manipulieren zu können, wurde der Editor erarbeitet, der ein fester Bestandteil von KC-FORTH ist.
Der Editor ist ein fester Satz von Worten, der in dem speziellen Teilwörterbuch (vocabulary) EDITOR zusammengefaßt ist.

Folgende Wörter stehen zur Verfügung:

EDIT (n – >) – unter Cursorsteuerung kann der Text in den Screen n geschrieben und verändert werden
CLEAR (n – >) – der Screen wird mit Leerzeichen überschrieben
MOVE (n1 n2 – >) – der Screen n1 wird in den Screen n2 kopiert
L (– >) – liste den aktuellen Screen aus
P (n – >) – die nachfolgende Zeile wird in die Zeile n des aktuellen Screens transportiert
H (n – >) – umspeichern der Zeile in den Speicherbereich PAD
R (n – >) – ersetzen der Zeile mit der in PAD stehenden
T (n – >) – Zeile auf Bildschirm darstellen
E (n – >) – Zeile wird mit Leerzeichen überschrieben
S (n – >) – an die Position n wird eine Leerzeile eingefügt
D (n – >) – löschen der Zeile

2.4. KC-spezifische Wörterbucherweiterungen

Die Fähigkeiten des KC85 gehen weit über das von fig-FORTH unterstützte Maß hinaus. Deshalb wurden in KC-FORTH eine Reihe von Worten implementiert, die die Nutzung dieser zusätzlichen Funktionen in FORTH-typischer Weise erlauben.

- Ein-/Ausgabesteuerung

SETIN SETOUT – Kanalumschaltung der Ein-/Ausgabegeräte
NORMIN NORMOUT – Umschaltung auf „normale" Ein-/Ausgabe (Tastatur, Bildschirm)
BUSIN BUSOUT – Byteein- bzw. -ausgabe von bzw. zu Peripheriebausteinen des Systems

- Speichermanipulation im Bildwiederholspeicher

I§ IC§ – Holen von Speicherinhalten auf den Stack
I! IC! – Speichern von Zahlen bzw. Bytes auf die Adresse im IRM

- Modulsteuerung

MODUL – Abfrage der aktuellen Hardwarekonfiguration (Lesen des Modultyps)
SWITCH – Programmieren eines Moduls

- Tonerzeugung

SOUND – Erzeugung eines Tones oder eines Akkords aus zwei Tönen

- Farb- und Cursorsteuerung

INK INKP – Vordergrundfarbe für Zeichen und Grafik einstellen
BLINK BLINKP – Blinkmodus für Zeichen oder Grafik umschalten
PAPER – Hintergrundfarbe einstellen
COLOR – Vorder- und Hintergrundfarbe einstellen
LOC – Positionieren des Cursors
WINDOW – Einstellen eines Fensters auf dem Bildschirm

Grafik

PIX – Bildpunkt setzen oder löschen
PLOT – Vektor (Linie) zeichnen oder löschen

Magnetbandarbeit

CSAVE CLOAD – Speichern/Lesen von Magnetbanddateien in Screens
R/W – blockweises Lesen oder Schreiben von Magnetbanddateien

Sonstiges

CAOS – Aufruf eines Unterprogramms des Betriebssystems CAOS mit Parameterübergabe
BYE – Übergabe der Steuerung an das Betriebssystem
DUMP – Hex-Dump eines Speicherbereiches

3. Arbeit mit FORTH im System KC85

Mit dem Modul M026 ist für das Kleincomputersystem KC85 des VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen die Programmiersprache FORTH verfügbar. Da der FORTH-Interpreter in einem ROM-Modul enthalten ist, entfällt vor dem Start das Einlesen von Kassette. Über das SWITCH-Kommando des Betriebssystems ist der BASIC-ROM des KC85/3 abzuschalten und der Modul M026 zuzuschalten. Danach wird der Interpreter durch Eingabe des Menüwortes FORTH gestartet.
Das KC-FORTH berücksichtigt alle Besonderheiten des KC85/2 und KC85/3. Dem Programmierer sind die Systemfunktionen in FORTH-typischer Art zugängig. Insbesondere die Modulverwaltung erlaubt die vollständige Nutzung unterschiedlicher Hardwarekonfigurationen. In FORTH ist z. B. problemlos die Verwendung auch von mehreren RAM-Modulen möglich. Im Grundgerät können somit maximal 136 KByte RAM insbesondere als Quelltext- und Datenspeicher eingesetzt werden.
Da der KC85 in seiner Grundausrüstung als Massenspeicher eine Magnetbandkassette besitzt, wurde aus Kompatibilitätsgründen im Speicher des KC eine virtuelle Diskette (RAM-Floppy) eingerichtet. Der Nutzer kann die Lage und Größe dieser virtuellen Diskette selbst festlegen.
Da über die Programmiersprache FORTH bisher sehr wenig Literatur zugängig ist, wird vom Hersteller zum Modul ein umfangreiches Handbuch mitgeliefert. Dieses Handbuch erläutert in den ersten 7 Kapiteln die Programmiersprache und ermöglicht das Erlernen der Programmiertechnik an einfachen und einleuchtenden Beispielen. Im Kapitel 8 wird ein kompletter Überblick über alle Worte des KC-FORTH mit Erklärung der Funktion und der Stack-Belegung gegeben.
Im Anhang sind die wichtigsten Systemdefinitionen und ein Verzeichnis der KC-FORTH-Worte enthalten.
Der Modul kann über den üblichen Vertriebsweg bei VEB Robotron-Vertrieb Magdeburg bzw. über die RVx-Betriebe bestellt werden und ist wie der KC85/3 für gesellschaftliche Bedarfsträger vorgesehen.

Literatur

/1/ Kleincomputer KC85. Beschreibung zu M026 FORTH. VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen 1987
/2/ Greene, R. L.: Faster FORTH, Byte 9 (1984) S. 127
/3/ Broody, L.: Programmieren in FORTH. Hansa-Verlag und Prentice-Hall International. München, Wien, London 1984
/4/ Zech, R.: Die Programmiersprache FORTH. Franzis-Verlag München 1984
/5/ Kühnel, C.: Erste Erfahrungen mit FORTH. Radio, Ferns., Elektron. Berlin 35 (1986) 9, S. 553

Modul M027 Development

Assemblerprogrammierung für KC 85/3

Klaus-Dieter Kirves
VEB Mikroelektronik „Wilhelm Pieck"
Mühlhausen

Der Kleinrechner KC 85/3 bietet neben der Programmierung mittels BASIC-Interpreter auch die Möglichkeit der U880-Maschinencode- oder Assemblersprache.
Bei der BASIC-Programmierung können bei weitem nicht alle Möglichkeiten des KC85-Systems ausgeschöpft werden. Assemblerprogramme bieten Vorteile in der Rechengeschwindigkeit, der Speicherverwaltung des Systems und der Steuerung peripherer Geräte besonders im Interruptbetrieb. Das Betriebssystem CAOS des KC 85/3 unterstützt dies durch eine große Anzahl vom Anwender nutzbarer Unterprogramme.
Ein wichtiger Anwendungsbereich der Assemblerprogrammierung sind Industrierobotersteuerungen und CAD/CAM-Systeme. Mit dem hier vorgestellten Development-Modul ist der KC 85/3 gut zum Erlernen der Assemblersprache, Erstellen und Testen von Programmen, auch für andere Rechner, geeignet. Die Beschäftigung mit der Assemblerprogrammierung bietet auch dem Anfänger bzw. bisherigen BASIC-Programmierer tieferen Einblick in die Arbeitsweise eines Mikrorechners.
Ebenso erweitert die Unterstützung von BASIC-Programmen durch Maschinenunterprogramme (z. B. für zeitkritische Probleme) den Einsatzbereich des Kleincomputers KC 85/3. Der Modul kann ebenfalls im Kleincomputer KC 85/2 genutzt werden.

Übersicht zum Modul M027 Development

Es handelt sich um einen 8KByte-Festwertspeichermodul, der funktionell mit dem 8K-USER-EPROM-Modul M025 übereinstimmt. Der Modul enthält in vier 2KByte-PROMs einen Zwei-Pass-Assembler mit Editor für die Z80-Sprachversion (entspr. UDOS- bzw. SCP-Assembler), einen Disassembler, einen Testmonitor sowie Treiberprogramme zur Druckeransteuerung über V24-Modul M003. Der Modul belegt den Speicherbereich von C000H bis DFFFH und kann über Kommando zu- und abgeschaltet werden.
Beim Einsatz im KC 85/3 muß nach dem Zuschalten des Developmentmoduls im rechten Modulschacht (CAOS-Kommando 'SWITCH 8 C1') der interne BASIC-ROM abgeschaltet werden, da dieser den gleichen Speicherbereich belegt. Das CAOS-Kommando dafür lautet 'SWITCH 2 0'.

Editor/Assembler

Editor und Assembler des Developmentmoduls bilden eine Einheit. Sie werden aus dem CAOS-Menü mit den Kommandos EDAS bei Kaltstart bzw. REEDAS bei Warmstart aufgerufen. Sie arbeiten mit einem eigenen Untermenü im gleichen Menükonzept wie das Betriebssystem. Es wird der verfügbare RAM-Speicherbereich ab Adresse 200H bis 3FFFH ohne Speichererweiterung oder bis 7FFFH mit 16K-RAM-Erweiterung (Modul M022) benutzt (Bild 1). Eine Begrenzung des Speicherbereichs ist möglich. Drei Viertel des verfügbaren Speichers sind für den Quelltext vorgesehen. Der restliche Teil dient als Speicher für die Symboltabelle des ersten Assemblerlaufs und für den übersetzten Maschinencode. Damit ist es je nach Umfang der Programmkommentare möglich, mit dem Grundgerät ohne RAM-Erweiterung Maschinenprogramme maximal etwa 2 KByte und mit RAM-Erweiterung maximal etwa 4 KByte zusammenhängend zu programmieren.

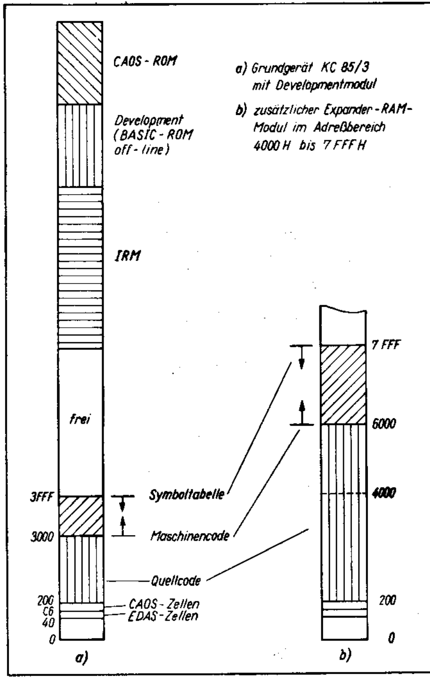


Bild 1
Speicherplatzaufteilung im KC 85/3 beim Einsatz des Assemblers des Development-Moduls M0 27

Der Assembler verarbeitet die Mnemonik der Z80-Assemblersprachversion. Tafel 1 zeigt alle zulässigen Pseudobefehle. Marken dürfen aus maximal 6 Zeichen bestehen. Kommentare werden durch Semikolon abgetrennt. Zahleneingaben können dezimal oder hexadezimal (mit ,H' gekennzeichnet) erfolgen.
Als Zeichen für den aktuellen Befehlszählerstand dient das Währungszeichen $ oder das Doppelkreuzzeichen #, diese werden bei relativen Sprüngen nicht eingegeben. Die Quellenzeilenlänge beträgt entsprechend Bildschirmformat des Kleincomputers 40 Zeichen. Tafel 2 gibt einen Überblick zu den Untermenükommandos.
Der Editor ist bildschirmorientiert, wobei alle Cursorfunktionen des Betriebssystems (Löschen, Einfügen, Zeilenlöschen, Bildschirmlöschen), erweitert um eine Tabulatorfunktion, verfügbar sind. Der Übergang von der Texteingabe ins Untermenü erfolgt mit der BREAK-Taste. Dabei wird über dem Menü der noch freie Speicherbereich angezeigt. Aus dem Menü kann der Text an der Stelle wieder erreicht werden, wo abgebrochen wurde; es kann an den Anfang oder das Ende des Textes gegangen werden. Eine weitere Möglichkeit ist, über eine Suchfunktion eine bestimmte Zeichenkette (z. B. eine Marke) anzuwählen. Beim Editieren ist es möglich, den Text bildschirmseitenweise vorwärts und rückwärts zu blättern und vorwärts zeilenweise zu rollen. Der Quelltext kann vollständig mit Dateityp ASM und Namen auf Kassette abgespeichert werden. Beim Wiedereinlesen von Kassette ist ein Einfügen in im Speicher stehende Texte möglich.
Beim Aufruf des Assemblers sind mehrere Optionen (Tafel 3) möglich. Die Übersetzung erfolgt in zwei Läufen. Im ersten Lauf werden die Symboltabelle (Markentabelle) und im zweiten Lauf der Maschinencode und das Listing erzeugt. Im zweiten Lauf erfolgt ebenfalls die Ausgabe eventueller Fehler. Es werden 11 Fehlertypen unterschieden.
Die Ausgabe des Listings erfolgt über Bildschirm oder – wenn vorhanden – über Drukker oder elektronische Schreibmaschine. Bei Bildschirmausgabe kann die Zeilenlänge auf 40 Zeichen begrenzt werden, um eine übersichtliche Darstellung zu erhalten. Das Maschinenprogramm wird, wenn keine ORG-Anweisung vorhanden ist, im Speicher ab Adresse 3000H bzw. 6000H bei Speichererweiterung abgelegt. Mittels einer ORG-Anweisung kann das Programm auf beliebige Adreßbereiche übersetzt werden. Ist der Zieladreßbereich während der Assemblierung belegt, so kann der Maschinencode auch sofort auf Kassette (als Dateityp COM) ausgegeben werden.
Mit den angegebenen Optionen ist es auch möglich, zwei oder mehrere Quellprogramme zu binden (Linken). Dazu wird zuerst die Gesamtmarkentabelle aufgebaut, indem die einzelnen Quellprogramme mit Option + übersetzt werden. In einem zweiten Durchlauf werden die Maschinenprogramme mit den Optionen 0 und 2 erzeugt. Eine andere Möglichkeit besteht darin, alle Systemadressen und wichtigen Codes in einem Quellprogramm mittels EQU-Anweisungen einmal zu vereinbaren und dieses Quellprogramm vor dem eigentlichen Quellprogramm zu übersetzen und so die Markentabelle aufzubauen. Somit kann bei allen Anwenderprogrammen mit den symbolischen Namen gearbeitet werden.
Weiterhin kann mit der Menüanweisung PRINT der Textpuffer angezeigt oder ausgedruckt werden. Durch die Angabe von zwei Parametern kann das Druckformat gewählt werden. Der erste Parameter bestimmt, wieviele Bildschirmzeichen als eine Textzeile ausgedruckt werden (z. B. 80 oder 120 Zeichen). Der zweite Parameter gibt an, wieviele Zeilen je Seite gedruckt werden sollen.

Disassembler

Zur Kontrolle selbsterstellter Programme und zur Analyse fremder Programme ist ein Disassembler sehr hilfreich. Der im Modul enthaltene Disassembler liefert die gleiche Mnemonik, die auch der Assembler verarbeitet. Als Besonderheit unterstützt er die Betriebssystemaufrufe über Unterprogrammnummer in folgender Form:

Tafel 1 Pseudobefehle

Mnemonik	Bedeutung
DEFB nn	Belegung einer RAM-Zelle mit dem Wert nn
DEFW nnnn	Belegung zweier Speicherzellen mit dem Wert nnnn
DEFM'XXX'	Belegung eines Speicherbereiches mit der ASCII-Zeichenkette xxx
EQU	Vereinbarung einer Marke mit einem Ausdruck (Zahl, Marke oder Verknüpfung)
ORG nnn	Festlegung der Adresse, auf welcher das Maschinenprogramm lauffähig ist, ohne Angabe gilt 3000H bzw. 6000H

Tafel 2 Anweisungen des Editor/Assembler-Menüs

Anweisung	Bedeutung
EXIT	Rückkehr ins Betriebssystem
TOP	Editieren ab Textanfang
BOTOM	Editieren ab Textende
EDIT	Editieren ab Austrittsstelle
FIND	Suchen einer Zeichenkette und Übergang in den Editiermodus
CLEAR	Löschen Quelltext
SAVE	Abspeichern Quelltext auf Kassette
LOAD	Einlesen Quelltext von Kassette
ASM	Aufruf des Assemblers
PRINT	Ausdruck des Quelltextpuffers
VERIFY	Kontrollesen der Kassettenaufzeichnung

Tafel 3 Optionen des Assemblers

Option	Bedeutung
+	beim 1. Assemblerlauf wird die Markentabelle an die bereits vorhandene „angehängt"
2	der 1. Assemblerlauf entfällt
L	es wird ein Assemblerlisting ausgegeben
P	(in Verbindung mit L) die Ausgabe erfolgt auf Drucker
B	die Länge der Ausgabezeile wird auf 40 Zeichen begrenzt
O	es wird der Maschinencode im RAM abgelegt
S	der Maschinencode wird auf Kassette ausgegeben

Tafel 4 Anweisungen des Testmonitors

Anweisung	Bedeutung
BREAK, SBRK	Vereinbarung eines Unterbrecherpunktes
GO	Start eines Anwenderprogrammes in Echtzeit
STEP	Abarbeitung eines Befehls
COPY, MOVE	Verschieben von Speicherbereichen
IN	Lesen eines Eingabeports
OUT	Ausgabe eines Bytes an einen Ausgabeport
DISPLAY	Ausgabe eines HEX-Dumps
ARITH	Hexadezimalarithmetik
FILL	Beschreiben eines Speicherbereiches
CHSUM	Berechnung der Prüfsumme (Signatur) über einen Speicherbereich
WORKRAM	Verlagerung der Systemarbeitszellen im RAM
MODIFY	Veränderung von Einzelbytes
EXIT	Rückkehr ins Betriebssystem

```
0000 CALL 0F003H (Systemaufruf)
0004 DEFB 004H (Unterprogrammnummer)
```

Die Ausgabe von Texten kann über das Unterprogramm Nr. 23H folgendermaßen erfolgen:

```
0000 CALL 0F003H
0003 DEFB 023H
0004 DEFM'Text'
0008 DEFB 00AH (Zeilenschaltung)
0009 DEFB 00DH (Wagenrücklauf)
000A DEFB 000H (Endekennzeichnung)
```

Beide Systemaufrufe werden vom Disassembler in der dargestellten Weise ausgegeben. Es ist auch möglich, das vom Disassembler erzeugte Quellprogramm nicht auf Bildschirm oder Drucker auszugeben, sondern als Datei vom Typ ASM auf Kassette abzuspeichern und mit Editor und Assembler weiterzubearbeiten.

Testmonitor

Neben den Hilfsmitteln für die Programmerstellung sind auch die zur Programmtestung sehr wichtig. Als Kernstück dienen in diesem Fall eine softwaremäßige Unterbrechungspunktsteuerung und eine interruptgesteuerte Einzelschrittprogrammabarbeitung. Tafel 4 zeigt einen Überblick zu den Anweisungen des Testmonitoruntermenüs. Der Testmonitor wird aus dem CAOS-Grundmenü mittels der Anweisung TEMO bzw. RETEMO aufgerufen.

Beim Start des zu testenden Programmes werden die CPU-Register aus bestimmten RAM-Zellen geladen. Damit ist es möglich, jederzeit die Anwenderregister zu beeinflussen und anzuzeigen. Beim Kaltstart des Testmonitors werden alle Anwenderregister mit Ausnahme des Stackpointers (SP) auf Null gesetzt. Der Stackpointer wird während der Abarbeitung eines Anwenderprogrammes auf einen Bereich unterhalb des Systemstacks gelegt. Alle Register können über Menüanweisung direkt gesetzt werden. Die Bezeichnung des entsprechenden Einzel- oder Doppelregisters oder CARRY- bzw. ZERO-Flags erfolgt mittels Kleinbuchstaben mit nachfolgendem Parameter. Der Start eines zu testenden Anwenderprogrammes kann auf zweierlei Arten erfolgen. Die erste Version ist der Start in Echtzeit mittels GO-Anweisung. Ein mit der BREAK-Anweisung vorher vereinbarter Unterbrechungspunkt wird eingetragen, d. h. auf die angegebene Adresse (erstes Byte eines Befehls) wird ein RST-38H-Befehl (Byte FFH) eingetragen. Wird während des Programmlaufs ein RST 38H erkannt, so werden die Anwenderregister im RAM gerettet, das ersetzte Byte ins Programm zurückgeschrieben, und der Testmonitor geht in die Registeranzeige. Hierbei werden alle Register (A, BC, DE, HL, SP, PC, IX, IY), die Flags und die Bytes als auch die Mnemonik des Folgebefehls angezeigt.
Bei der zweiten Möglichkeit erfolgt die Abarbeitung interruptgesteuert. Der Start erfolgt hierbei mit der STEP-Anweisung. Die Anwenderregister werden wiederum geladen. Es wird jedoch der CTC-Kanal 0 (entspricht dem Tonkanal 2) des KC 85/3 als Zeitgeber so programmiert, daß nach Abarbeitung eines Anwenderbefehls ein Interrupt ausgelöst wird. Wurde vorher mit der SBRK-Anweisung ein Haltepunkt vereinbart, werden solange einzelne Anwenderbefehle abgearbeitet, bis der aktuelle Befehlszählerstand dem Haltepunkt entspricht. Danach wird in die Registeranzeige gesprungen. Wurde kein Haltepunkt vereinbart, so erfolgt dies sofort nach dem ersten Befehl. Aus der Registeranzeige kann mit der ENTER-Taste die Einzelschrittverarbeitung fortgesetzt werden. Bei der Betätigung der CURSOR DOWN-Taste wird ein Haltepunkt auf den folgenden Befehl (vgl. SBRK) gesetzt und die Einzelschrittverarbeitung gestartet. Damit ist es möglich, Programmschleifen und Unterprogramme komplett abzuarbeiten. Dies gilt natürlich auch für die Systemunterprogrammaufrufe (vgl. Disassembler). Die Abarbeitung ist etwa um den Faktor 120 langsamer als Echtzeitverarbeitung. Sie kann jederzeit mittels BRK-Taste angehalten werden. Es erfolgt dabei ein Übergang in die Registeranzeige.
Werden die Anweisungen STEP oder GO ohne Parameter aufgerufen, so wird auf der Adresse gestartet, auf welche der Anwender-PC zeigt. Es ist damit leicht möglich, den Schrittbetrieb und auch den Testmonitor zu verlassen und anschließend an der gleichen Stelle wieder fortzusetzen. Werden nicht benötigte RAM-Speicherbereiche mit der FILL-Anweisung auf FFH gesetzt, so kann die Gefahr eines Programmabsturzes verringert werden, da beim „Auflaufen" auf einen RST-38H-Befehl der Testmonitor die Steuerung übernimmt.
Mit den IN- und OUT-Anweisungen können leicht Peripherieschaltkreise programmiert und getestet werden. Die Ein- und Ausgabebefehle arbeiten mit 16-Bit-Adressen. Die ARITH-Anweisung ermittelt neben Summe und Differenz zweier 16-Bit-Hexadezimalzahlen auch den Sprungoffset für relative Sprungbefehle.
Zur Überwachung von Speicherbereichen auf Veränderungen und zur Kontrolle von EPROMs und Kassettenaufzeichnungen dient die CHSUM-Anweisung, welche eine Signatur über einen Speicherbereich bilden kann.

Druckertreiberprogramme

Eine wesentliche Erleichterung bei der Assemblerprogrammierung bietet die Möglichkeit der Protokollierung des Assemblerlistings oder der Testabarbeitung über einen

(Fortsetzung auf S. 249)

Grafische Bildschirmansteuerungen für Kleincomputer

Matthias Schreiber
VEB Robotron Meßelektronik „Otto Schön“
Dresden

Die Leistungsfähigkeit eines Computers wird entscheidend von seiner Bildausgabe mitbestimmt. Wichtigste Parameter bei dieser Beurteilung sind neben anderen die Auflösung und die Geschwindigkeit der Bildausgabe. Für Kleincomputeranwendungen stellt die Verwendung eines handelsüblichen Fernsehgerätes mit einer maximalen Auflösung von 512×256 Bildpunkten eine relativ leistungsfähige und ökonomische Lösung dar, obwohl heute schon für CAD/CAM-Anwendungen Terminals mit einer Auflösung von 4096×4096 Bildpunkten eingesetzt werden. Der folgende Beitrag soll die wichtigsten Bildausgabeverfahren sowie einige Realisierungsmöglichkeiten des Raster-Scan-Verfahrens vorstellen.

Bildausgabeverfahren

Zur Darstellung grafischer und alphanumerischer Informationen auf einem Terminal werden im allgemeinen drei Verfahren, das Vektor-Refresh-, das Speicher- und das Raster-Scan-Verfahren verwendet.
Das Vektor-Refresh-Verfahren, das in einer Bilddatei die Endpunkte der Linien abspeichert, die auf dem Bildschirm dargestellt werden sollen, und das Speicherverfahren, bei dem die Bilddaten in einer Speicherbildröhre abgespeichert werden, sind auf Grund ihres komplizierten Aufbaus und der aufwendigen Ansteuerung nur für kommerzielle Terminals anwendbar. Hier können ihre speziellen Vorteile, die Darstellungsmöglichkeit dynamischer Bewegungsabläufe und geringer Bildspeicherbedarf des Vektor-Refresh-Verfahrens sowie die sehr hohe Auflösung des Speicherverfahrens, voll ausgenutzt werden.

Beim Raster-Scan-Verfahren wird als Bildschirm eine Kathodenstrahlröhre verwendet, die in ihrem Aufbau einer Fernsehbildröhre entspricht. Das auf dem Bildschirm auszugebende Bild wird in einem Bildspeicher als Bit-Map (Bit-Plan) abgespeichert. Hierbei ist das Bild in eine Matrix von Pixeln (Bildpunkte) aufgeteilt, denen jeweils eine Speicherzelle des Bit-Map-Speichers (Bildspeicher) zugeordnet ist. Ein Display-Controller (Bildschirmansteuerung) liest den Bildspeicher aus und wandelt die gespeicherte Information in elektrische Signale um, die den Elektronenstrahl während der Zeilenablenkung ein- und ausschalten, so daß das Bild als Punktmuster auf dem Bildschirm erscheint.
Bei der Realisierung von Bildschirmansteuerungen nach dem Rasterverfahren gibt es verschiedene Möglichkeiten einer Grafikausgabe. Dabei wird bei den im folgenden beschriebenen Realisierungsmöglichkeiten, auf Grund des Zeilenflimmerns, das bei Verwendung des Zeilensprungverfahrens (Interlace) auftritt, grundsätzlich nur das Vollbildverfahren (Non interlace) angewendet. Aus ökonomischen Gründen wird auf die Verwendung handelsüblicher Fernsehgeräte orientiert, obwohl die beschriebenen Verfahren grundsätzlich auch für industrielle Terminals zutreffen, die nach dem Rasterverfahren arbeiten.
Nach der CCIR-Norm besteht ein Fernsehbild aus 625 Zeilen mit einer Zeilenfrequenz von 15625 Hz. Daraus ergibt sich eine Bildwechselfrequenz von 25 Hz. Da nach dem Halbbildverfahren gearbeitet wird, ergibt das eine Halbbildwechselfrequenz von 50 Hz. Zum Darstellen der Zeichen auf dem Bildschirm wird aus obengenannten Gründen auf das Halbbildverfahren verzichtet, und statt dessen 2 Bilder mit der Hälfte der Fernsehzeilen erzeugt. Damit stehen nur noch 312 Zeilen zur Bildausgabe zur Verfügung. Auf Grund der notwendigen Überschreibung des Bildes über den Bildschirmrand und der nichtnutzbaren Randzeilen (relativ starke Verzerrungen und Unschärfe an den Rändern) sind von diesen nur maximal 256 Zeilen (bei älteren Geräten auch nur 200 Zeilen) für eine Zeichenausgabe nutzbar. Gleiches gilt auch für die Auflösung in horizontaler Richtung. Hier kann bei Ausgabe über HF mit einer Auflösung von maximal 512 Bildpunkten gerechnet werden. Um eine Austauschbarkeit der Fernsehgeräte zu gewährleisten, sollten die Randzonen allerdings etwas größer eingeplant und die Bildschirmansteuerung zwecks besserer Zeichenerkennbarkeit für eine Auflösung von 320×200 Bildpunkten konzipiert werden. Erst bei einer Direktansteuerung der Bildröhre durch die Bildschirmansteuerung ohne den Umweg über Modulation und Demodulation ist eine höhere Auflösung sinnvoll.

Symbolgrafik

Eine optimale Zeichendarstellung auf dem Bildschirm wird durch seine Aufteilung in Zeichenfelder, die aus einer 8 × 8-Punktmatrix bestehen, erreicht. Durch die Wahl dieser Punktmatrix, der Pixelfrequenz von ca. 7,5 MHz und der Anzahl der darzustellenden Punkte pro Zeile ist es auf Grund des Bildschirmformates möglich, im Grafikmode Quadrate oder Kreise ohne Korrekturfaktor zu erzeugen. Damit sind auf dem Bildschirm 25 Zeichenzeilen mit je 40 Zeichen je Zeile darstellbar. Bild 1 zeigt das verwendete Bildformat. Jeder Zeichenposition auf dem Bildschirm wird eine Adresse im Bildwiederholspeicher in steigender Reihenfolge zugeordnet. Der Bildposition 0 (Zeile 0, Spalte 0) entspricht die Anfangsadresse, und der Position 999 (Zeile 24, Spalte 39) die Endadresse des Bildwiederholspeichers. Demnach ist die Darstellung von maximal 1000 Zeichen, das entspricht einem Bildwiederholspeicher mit einer Größe von etwa 1 K Byte, möglich. Um das entsprechende Farbattribut für jedes Zeichen speichern zu können, muß ein Farbattributspeicher gleicher Größe parallel dazu angeordnet werden. Bild 2 zeigt den Bildaufbau für das beschriebene Prinzip. Die Information wird für alphanumerische Zeichen auf 7 Fernsehzeilen dargestellt, die 8. Zeile dient zur Trennung der Zeichenzeile. Hier ist es allerdings zwecks Übersichtlichkeit und guter Lesbarkeit des Bildinhaltes günstig, weitere Leerzeilen einzufügen. Gute Systeme überlassen es dem Nutzer, zu entscheiden, wieviel Leerzeilen eingefügt werden sollen. Die im Bild 2 dargestellte Punktmatrix wird einem Zeichengenerator entnommen. In diesem werden für ein darzustellendes Zeichen acht Speicherzellen benötigt. Der ASCII - Code des darzustellenden Zeichens stellt den höherwertigen Teil der Zeichenadresse dar, die drei niederwertigen Adreßbits werden dem Zeilenzähler entnommen, der die aktuelle Fernsehzeile für die Zeichen enthält. Bild 3 zeigt die möglichen Kombinationen am Zei-

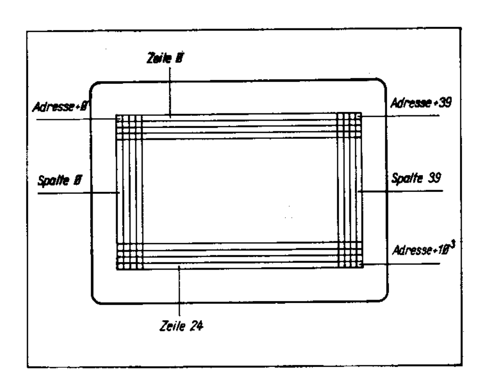


Bild 1 Bildformat

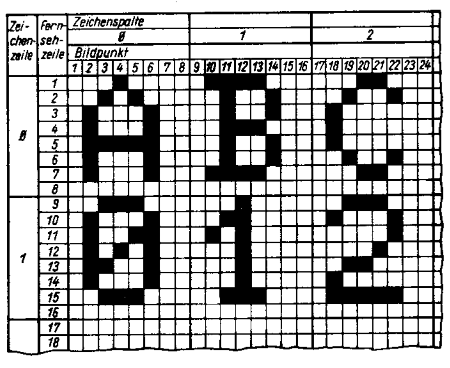


Bild 2 Bildaufbau

(Fortsetzung von S. 248)
Drucker. Der Modul enthält zwei Druckertreiberprogramme, abgestimmt auf den V24-Modul M003. Beide Treiberprogramme gestatten die Ausgabe des Assemblerlistings sowie die parallele Mitprotokollierung aller Bildschirmausgaben.
Der Treiber V24K6311 ist vorrangig für die Matrixdrucker K6311 und K6312 vorgesehen. Er initialisiert bei diesen Druckern die Formulargröße für Leporello-Papier und arbeitet mit 1200 Baud.
Der Treiber V24K6313 arbeitet mit 9600 Baud. Er kann z.B. die Drucker K6313, K6314, K6304, sowie die elektronischen Schreibmaschinen S6005 und S6010 und Druckgeräte mit gleicher Schnittstelle bedienen.
Ein Anschluß von anderen Druckern mit anderen Schnittstellen ist vom System her auch möglich, z. B. durch Nachladen einer Treiberroutine und ggf. Einsatz eines speziellen Ausgabemoduls.

Zeile	Darstellung Bildpunkt								Zeichengeneratorinhalt BIN								HEX	Zeichengeneratoradresse ASCII-Code 41H								LC		
	1	2	3	4	5	6	7	8	D7	D6	D5	D4	D3	D2	D1	D0	10	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1				■					0	0	0	1	0	0	0	0	10	0	1	0	0	0	0	0	1	0	0	0
2			■		■				0	0	1	0	1	0	0	0	28	0	1	0	0	0	0	0	1	0	0	1
3		■				■			0	1	0	0	0	1	0	0	44	0	1	0	0	0	0	0	1	0	1	0
4		■				■			0	1	0	0	0	1	0	0	45	0	1	0	0	0	0	0	1	0	1	1
5		■	■	■	■	■			0	1	1	1	1	1	0	0	7C	0	1	0	0	0	0	0	1	1	0	0
6		■				■			0	1	0	0	0	1	0	0	44	0	1	0	0	0	0	0	1	1	0	1
7		■				■			0	1	0	0	0	1	0	0	44	0	1	0	0	0	0	0	1	1	1	0
8									0	0	0	0	0	0	0	0	00	0	1	0	0	0	0	0	1	1	1	1

Bild 3 8 × 8 Matrix und Zeichengeneratororganisation

Bild 5 Viertelgrafikzeichen

Zeichen																
Code HEX	20	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	FF
Code DEZ	32	176	177	178	179	180	181	182	183	184	185	186	187	188	189	255

chen „A". Wird für den Zeichengenerator ein Speicher mit einer Speicherkapazität von 2 K Byte eingesetzt, können maximal 256 Zeichen dargestellt werden. Der Zeichenvorrat könnte beispielsweise 96 alphanumerische und 128 Grafikzeichen betragen. 32 Codierungen werden meist nicht benutzt, da sie für Steuerzeichen reserviert sind. Bei Nutzung der Grafikzeichen ist es allerdings nicht erforderlich, zusätzliche Leerzeilen einzufügen, da in diesem Fall keine zusammenhängenden Bilder erzeugt werden können. Sind diese Grafikzeichen günstig gewählt, so läßt sich ein großer Teil grafischer Probleme lösen. Diese sogenannte Quasigrafik hat dazu noch den Vorteil einer relativ hohen Bilderzeugungsrate, da nur 1 K Byte Bildwiederholspeicher vom Prozessor verwaltet werden muß. Natürlich ist es nicht möglich, allen Anwendungsfällen zu genügen, da ja der Zeichensatz begrenzt ist. Beispiel für die eben beschriebene Ansteuerung sind die Bildschirmansteuerungen der Kleincomputer KC 85/1 und KC 87. Über die Leistungsfähigkeit dieser Quasigrafik geben einige für diese Kleincomputer erarbeitete Anwenderprogramme Auskunft. Mit den im Bild 5 dargestellten Grafikzeichen ist es möglich, eine sogenannte Viertelgrafik zu erzeugen. Hierbei wird der Bildschirm in ein Punktraster von 80×48 Punkte unterteilt. Mit dieser Viertelgrafik läßt sich beispielsweise der prinzipielle Verlauf, allerdings kein punktgenauer, mathematischer Funktionen darstellen. Des weiteren ist es mit einer quasigrafischen Bildausgabe möglich, Bilder zu erzeugen, Tabellen aufzustellen, Symbole darzustellen, Balkendiagramme anzuzeigen und vieles andere mehr.

Programmierbarer Zeichengenerator

Eine wesentlich bessere Möglichkeit der grafischen Darstellung ergibt sich, wenn der Zeichengenerator frei programmierbar ist. Hier läßt sich der jeweils benötigte Kurvenzug in den Zeichengenerator eintragen. Damit kann prinzipiell jeder Bildpunkt des Bildschirmes angesprochen werden. Den gesamten Bildschirm mit beliebigen verschiedenen Grafikzeichen zu belegen ist natürlich auch hier nicht möglich, da bei einer 8-Bit-Organisation des Bildspeichers die Anzahl der möglichen Grafikzeichen auf 128, oder wenn auf die alphanumerischen Zeichen verzichtet wird, auf 256 begrenzt ist. Da dieser Fall aber sehr selten eintritt, sind bei dieser Variante einer Bildschirmansteuerung die grafischen Darstellungsmöglichkeiten als sehr gut zu bezeichnen, wobei der benötigte Speicherplatz immer noch relativ klein bleibt. Für das oben genannte Bildformat werden 1 K Byte für den Bildwiederholspeicher, 1 K Byte für den Farbattributspeicher und 2 K Byte für den Zeichengenerator benötigt. Demzufolge werden bei dieser Variante insgesamt 4 K Byte des Adreßbereichs des Prozessors für die Bildausgabe verwendet, da ja nun der Zeichengenerator ebenfalls adressiert werden muß. Für mehrfarbige Darstellungen haben die bisher beschriebenen Bildschirmansteuerungen einen Nachteil. Das Farbattribut wird hier für ein ganzes Zeichen (Vordergrund- und Hintergrundfarbe) definiert. Treffen sich nun innerhalb eines Zeichens beispielsweise drei Farbflächen, so entsteht für eine der drei Farben an dieser Stelle ein Fehler, da nur zwei Farben innerhalb eines Zeichens auftreten können. Bild 6 zeigt dieses Problem.

Der Programmaufwand für diese Art der Grafikausgabe ist höher als im vorigen Beispiel, so daß ein BASIC-Programm die zeitlichen Anforderungen unter Umständen nicht mehr erfüllen kann. Hier kann man sich aber mit Maschinenprogrammroutinen helfen. Diese Art der Bildschirmansteuerung wird beispielsweise in einer Version des PC 1715 eingesetzt. Die mit dieser Bildschirmansteuerung erzeugbaren Grafikbilder entsprechen weitgehendst denen einer Vollgrafik, können aber nicht deren Vielfältigkeit bei gleichzeitig darstellbaren Figuren erreichen. Gegenüber der Variante mit festem Zeichengenerator stellt diese Grafikausgabe einen relativ großen Qualitätssprung dar.

Vollgrafik

Hochwertige grafische Darstellungen lassen sich nur mit einer vollgrafischen Bildschirmansteuerung erreichen. Der Aufwand, der im Vergleich zu den vorigen Beispielen getrieben werden muß, ist wesentlich größer. Das betrifft insbesondere den Bildwiederholspeicher, da jeder Punkt des Bildschirmes einer Speicherzelle des Bildwiederholspeichers entspricht. Soll die Darstellung dazu noch farbig sein, so werden für jeden Bildpunkt bei 8farbiger Darstellung drei Speicherzellen und bei 16farbiger sogar vier Speicherzellen benötigt. Für das gewählte Beispiel mit einer Auflösung von 320 × 200 Bildpunkten und einer 16farbigen Darstellung werden 256 000 Speicherzellen, das sind rund 32 KByte, benötigt. Wird ein solch großer Bildwiederholspeicher im Adreßbereich eines 8 Bit-Prozessors, der einen Speicherbereich von 64 KByte direkt adressieren kann, untergebracht, so belegt dieser schon die Hälfte des insgesamt direkt adressierbaren Speicherbereichs. Deswegen wird der Bildwiederholspeicher außerhalb des Systemspeichers angeordnet. Das erfordert, daß die Bildschirmansteuerung neben dem Bildaufbau auch die Verwaltung des Bildwiederholspeichers übernehmen muß. Für einen Bildwiederholspeicher in der Größenordnung von 32 KByte und mehr, ist es aus ökonomischen Gründen nicht mehr möglich, statische RAM-Schaltkreise zu verwenden. Dieser 32 KByte-RAM würde 64 Schaltkreise mit einer Organisation von 1K×4Bit (U 214, U 224) benötigen, während bei Verwendung des 16 K × 1 Bit dynamischer RAM U256 nur 16 Stück benötigt werden. Günstiger ist es, hier den 64 K × 1 Bit dynamischer RAM U 2164 einzusetzen, da dann nur 8 Schaltkreise benötigt werden, und es könnte außerdem ein zweites Grafikbild abgespeichert werden. Der Einsatz dynamischer RAM-Schaltkreise bedeutet aber, daß die Bildschirmansteuerung das Auffrischen der Bilddaten im Speicher ebenfalls übernehmen muß. Dadurch steigt der Aufwand sehr stark an, so daß diskret aufgebaute Lösungen mit diesem Leistungsum-

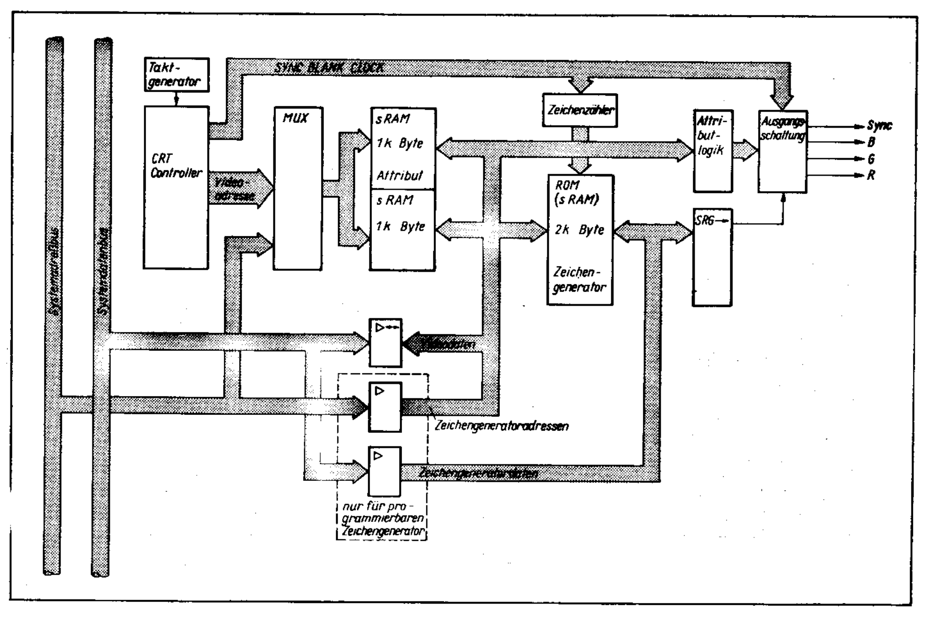


Bild 4 Blockschaltbild einer Bildansteuerung mit Zeichengenerator

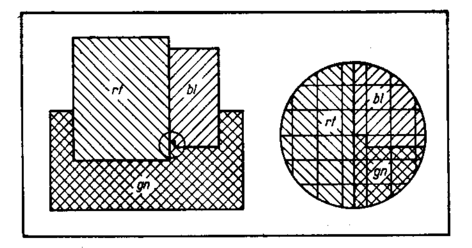


Bild 6 Farbfehler bei drei zusammentreffenden Farbflächen

fang für Kleincomputeranwendungen nicht in Frage kommen.
Natürlich gibt es auch Kleincomputer mit einer vollgrafischen Bildschirmansteuerung, die eine diskrete Lösung verwenden, aber dafür auf den möglichen Komfort verzichten; wie zum Beispiel beim KC 85/2 und KC 85/3.
Bei solcher Lösung ist der Bildwiederholspeicher im Arbeitsspeicher des Computers angeordnet, so daß der Prozessor die Verwaltung und Auffrischung übernehmen muß. Dabei müssen einige Kompromisse eingegangen werden. So wird der Anwenderspeicher um die Größe des benötigten Bildwiederholspeichers verringert. Das kann unterschiedlich sein, je nachdem ob im alphanumerischen oder im Grafik-Mode gearbeitet wird. Zum anderen ist es für ein störungsfreies Bild erforderlich, daß während der Bildausgabe der Zugriff des Prozessors auf diesen Speicherbereich verhindert wird. Da der Prozessor nun nur noch in den Austastlücken auf den Bildwiederholspeicher zugreifen kann, im anderen Fall müßte ein WAIT-Signal für den Prozessor erzeugt werden, kann ein solcher Zugriff bis zu 52 μs dauern. Im Mittel kann mit einer Bildspeicherzugriffszeit von 20 μs gerechnet werden. Das ist etwa die siebenfache Zeit eines normalen Speicherzugriffs. Da das die Bildaufbaurate ungemein verlängert, ist es unter Umständen notwendig, auch während der Bildausgabe auf den Bildspeicher zuzugreifen, was dann natürlich mehr oder weniger starke Störungen auf dem Bildschirm hervorruft.
Um eine komfortable und schnelle Grafik auch mit Kleincomputern realisieren zu können, ist der Einsatz eines hochintegrierten Schaltkreises für die Bildschirmansteuerung unbedingt nötig. Erst so kann ein Kleincomputer mit einer Grafik ausgerüstet werden, die leistungsfähig und kostengünstig ist und außerdem eine hohe Zuverlässigkeit besitzt. Bild 7 zeigt eine Lösungsvariante einer Bildschirmansteuerung mit einem Grafik-Display-Controller (GDC).
Dieser GDC gehört zur Gruppe der hochintegrierten Peripherie-Bausteine. Entstanden sind sie aus den CRT (Cathode-Ray-Tube)-Controllern, wie sie in den ersten beiden Beispielen beschrieben wurden, die dort noch diskret realisiert werden mußten und bei denen der Prozessor die Manipulation des Bildspeichers übernahm. Um den Prozessor von dieser Arbeit zu entlasten, wurden die ersten GDC entwickelt, die durch eingebaute Vektorgeneratoren selbständige Zeichenoperationen (Drawing) durchführen können. Neben dieser Funktion kann der GDC noch eine Reihe weiterer Aufgaben erfüllen:

- Verwaltung des vom Rechnerhauptspeicher getrennten Bildspeichers sowie Bereitstellung der benötigten Refreshadresse.
- Organisation der Übertragung von Befehlen und Parametern vom Prozessor in die GDC-Register sowie des Datenaustausches zwischen Prozessor und Bildspeicher.
- Bereitstellung der Synchronisations- und Austastsignale für das Videointerface.
- Ausführung von Zeichenoperationen (Drawing) für die meistgebrauchten grafischen Grundformen (Primitives): Linie, Rechteck, Kreisbogen, Punkt.

Die Verwendung eines solchen hochintegrierten Interface-Schaltkreises vereinfacht den Hardwareaufwand für komfortable Grafiksysteme erheblich.
Zusammenfassend kann festgestellt werden, daß schon mit einer relativ einfachen Bildschirmansteuerung, wie sie im ersten Beispiel beschrieben wurde, recht gute Ergebnisse bei einer grafischen Bildausgabe erreicht werden können. Für etwas anspruchsvollere Anwender ist jedoch ein programmierbarer Zeichengenerator unbedingt zu empfehlen, da damit Ergebnisse erreicht werden, die einer Vollgrafik nahekommen. Hochleistungsfähige Vollgrafik ist zur Zeit noch nicht für Kleincomputer realisierbar, was sich aber in nächster Zukunft mit der Bereitstellung entsprechender Interface-Schaltkreise ändern wird.

Matthias Schreiber (35) absolvierte von 1976 bis 1982 ein Fernstudium an der TU Dresden über Informationstechnik. 1982 diplomierte er mit einer Arbeit über Meßverfahren für Schaltzeiten schneller digitaler Schaltkreise. Seit 1982 ist Matthias Schreiber im Bereich Forschung und Entwicklung des VEB Robotron Meßelektronik „Otto Schön" Dresden tätig. Er beschäftigt sich zur Zeit mit Hardwareentwicklung von Kleincomputern, insbesondere mit Bildschirmansteuerungen.

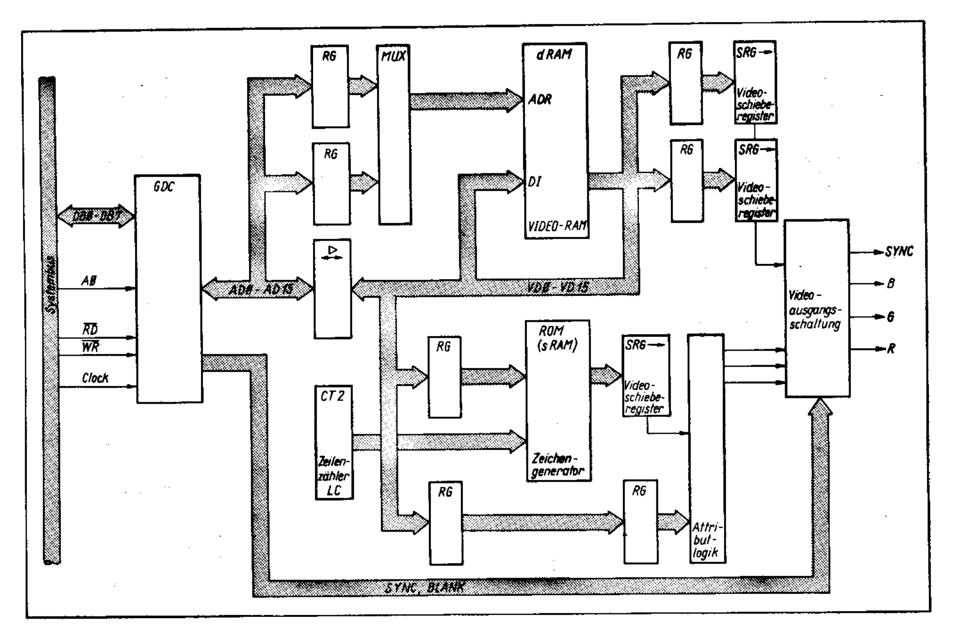


Bild 7 Blockschaltbild einer vollgrafikfähigen Bildausgabe mit Grafik-Display-Controller

Literatur

/1/ Mohr, A.: Entwicklungsstufen bei Grafikterminals, Der Elektroniker 23 (1984) 9, S. 59–61
/2/ Volk, A.: Ansichts-Sache, Elektronikpraxis (1985) 3, S. 16–25
/3/ VSDD steuert Bildschirm und Speicher. Elektronik (1985) 13, S. 79–83
/4/ Puchta, J.: Intelligenter Controller für Bit-Map-Grafik. Elektronik (1986) 2, S. 39–43
/5/ Birch, N.; Eckelmann, P.: CMOS-Baustein vereinfacht den Entwurf von Grafikterminals. Elektronik (1986) 5, S. 106–110
/6/ Strupat, A.: Grafikprozessor als zentraler Baustein für Dokumentationssysteme. Elektronik (1986) 5, S. 61–67
/7/ Gandhi, S.: Grafik-Coprozessor verwaltet 4-MByte-Bildspeicher. Elektronik (1986) 13, S. 60
/8/ Sterl, H.; Kurze, D.: Bildschirmansteuerung mit Einchip-Mikrorechner. Radio, Ferns., Elektron., Berlin 33 (1984) 1, S. 10–13
/9/ Kühnel, C.: Grafische Darstellungen auf dem Bildschirm, Radio, Ferns., Elektron., Berlin 33 (1984) 6, S. 290–293
/10/ Bogatz, A.: Videocontrollerschaltung. Radio, Ferns., Elektron., Berlin 34 (1985) 8, S. 500–502
/11/ Dünow, H.-P.; Halwass, M.: Grafikansteuerung für K 1520. Radio, Ferns., Elektron. 35 (1986) 10, S. 625–627
/12/ Kinder, F.; Schufft, M.: Bildschirmansteuerung für K 1520. Radio, Ferns., Elektron. 35 (1986) 10, S. 627–629

Literatur

/1/ Müller, S.: Einchipmikrorechner U883 interpretiert Tiny-MPBASIC. Radio, Ferns. Elektron., Berlin 34 (1985) 3, S. 143
/2/ Dugnus, G.: Neue Festwertspeicherbauelemente. Radio, Ferns., Elektron., Berlin 34 (1985) 11, S. 691
/3/ Müller, S.: Programmieren mit BASIC. REIHE AUTOMATISIERUNGSTECHNIK, Band 216. VEB Verlag Technik, Berlin 1985

☒ KONTAKT ☎

VEB Mikroelektronik „Karl Marx" Erfurt, Applikation Bauelemente, Abt. CEE, Rudolfstr. 47, Erfurt, 5010; Tel. 5 10 76 App. 40, (Koll. Möller)

LoRes-Plot-Programm mit Quasigrafik-Qualitäten

Christian Hanisch
Technische Universität Dresden,
Sektion Wasserwesen

Das LoRes-Plot-Programm DESIGN (LoRes = Low Resolution) ist ein selbständiges Druckprogramm, das maximal 900 x,y-Wertepaare aus einer ASCII-SDF-Datei unter Ausnutzung der quasigrafischen Möglichkeiten des Druckers SD 1152 auf das Format A3 (12-Zoll-Blattformat) als Kurve mit wählbarem Maßstab x:y ausgibt. Die Erstellung der Eingabedatei kann in einem vorgelagerten Nutzerprogramm oder mit Softwarewerkzeugen (DataStar, dBASE, WordStar usw.) erfolgen.
Das geforderte ASCII-SDF-Format ist identisch mit dem Format, das man mit dem BASIC-Statement

```
10 WRITE # 1, X, Y bzw.
20 WRITE # 1, A¤
```

auf Diskette erhält.

Die Eingabedaten der maximal fünf in einem Plot zu realisierenden LAYOUTS (Kurven) müssen als x,y-Tupel-Folgen nacheinander vorliegen. Signifikante Trennzeichen zwischen den Zahlen sind das Komma mit beliebig vielen Vor- oder Nachblanks sowie ⟨CR⟩, ⟨LF⟩ oder ⟨CRLF⟩. Die Abgrenzung der einzelnen LAYOUTS in der ASCII-SDF-Datei geschieht durch ein ASCII-Alpha- oder Sonderzeichen (z. B. +, *, #, x, %), das dort als Begrenzungszeichen fungiert und beim Drucken zur Kurvendarstellung benutzt wird. Die Form der Eingabedatei ist damit z. B.:

x, y, [x, y, ...], ...,* ⟨CRLF⟩

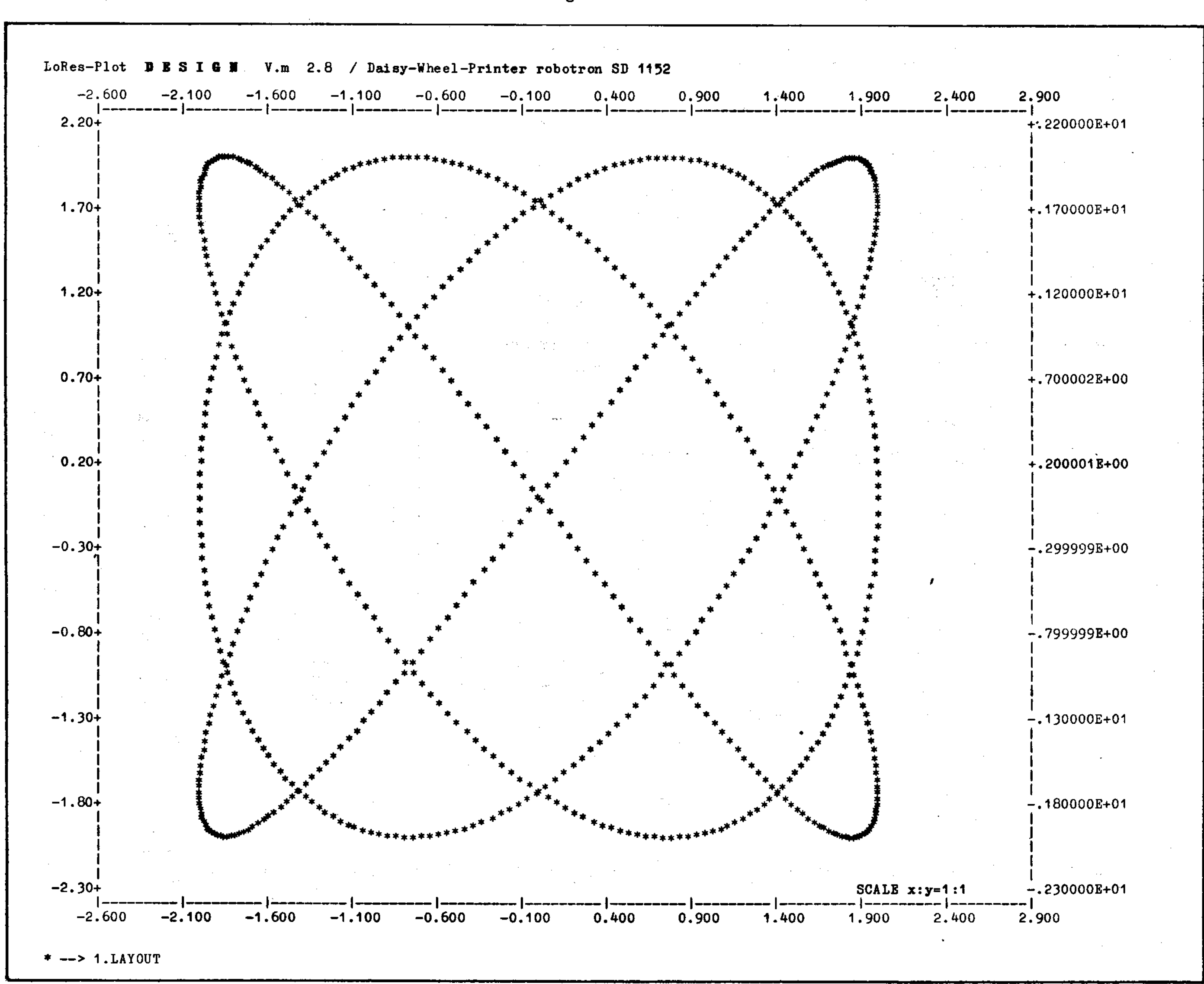
Die Formatierung des Plots auf die Größe eines A3-Blattes erfolgt automatisch. Standardmäßig wird der Maßstab x:y=1:1 realisiert. Darüber hinaus kann ein Maßstab x:y=1:a festgelegt werden bzw. wird vom Programm in den meisten Fällen aufgrund der Formatierung – ursächlich bestimmt durch die Werte- und Definitionsbereiche der LAYOUTS – ein zweiter Maßstab x:y=1:a (a⟨⟩1) angeboten.
Während eines Programmlaufes können mehrere Plots eventuell mit verschiedenen Maßstäben erzeugt werden.

Die Formatierung gewährleistet, daß die an die Koordinatenachsen im Abstand von einem Zoll ausgeschriebenen Werte keine durch das Ausgabeformat (x-Achse: ####.###, linke y-Achse: ###.##) gerundeten Zahlen sind. An der rechten y-Achse wird aus Platzgründen standardmäßig das E-Format gedruckt.
DESIGN gestattet weiterhin die Ausgabe mit absolutem Maßstab, wobei die Bildgröße der LAYOUTS das A3-Blattformat nicht überschreiten darf. Der absolute Maßstab setzt die Dimension Länge mit der Einheit Zentimeter in den Eingangsdaten voraus. Die Koordinatenachsen werden sowohl mit der Einheit Zentimeter als auch Zoll beschriftet. Die Positionierung der x,y-Werte auf dem A3-Blatt erfolgt mit einer Auflösung von 1/60 Zoll horizontal und 1/48 Zoll vertikal.
DESIGN eignet sich zur quasigrafischen Darstellung von Approximations- und Interpolationsproblemen. Aber auch die Darstellung von Ganglinien, Trendkurven usw. ist gut möglich. Für die praktische Anwendung sind 100 Kurvenpunkte für ein LAYOUT ausreichend. Der Verlauf der Kurve kann dann schon gut erkannt werden.

✉ KONTAKT ☏

Technische Universität Dresden, Sektion Wasserwesen, Bereich WE, Mommsenstraße 13, Dresden, 8027; Tel. 23 26 118

Bild 1 Beispielausdruck



Komponenten von CAD-Arbeitsplätzen

Christian Haas
VEB Zentrales Projektierungsbüro
der Textilindustrie

In starkem Maße werden in verschiedensten Bereichen interaktive grafische Arbeitsplätze auf 16-Bit-PC-Basis eingesetzt. Dabei stellt das Gebiet der reinen Zeichnungserstellung nur einen Gesichtpunkt dar. Es tritt vielmehr eine Verflechtung mit anderen Bereichen auf, die den Innovationsschub, der durch CAD realisiert wird, entscheidend beeinflußt. Diese Bereiche bestehen im wesentlichen aus

- dem Berechnen und Gestalten
- der geometrischen Modellierung
- der Modellierung von Funktions- und Bewegungsabläufen
- der Arbeitsplanung (teilweise als eigenständiges Gebiet dargestellt)
- der Erstellung und Aktualisierung von Stücklisten
- der NC-Programmierung
- der Herstellung von Dokumentationsunterlagen /1/.

Die Vielzahl der Aufgaben erfordert die Anwendung geeigneter CAD-Lösungen, wobei als CAD-Lösung eine Einheit von Hard- und Software verstanden wird. Nicht jede Aufgabenstellung kann und soll von einem System erfüllt werden. Der Anwendungsfall entscheidet über die Struktur der CAD-Lösung, also über die Struktur von Hard- und Software. Aus diesem Grund spielen Einsatzvorbereitung und Einführungsstrategie eine entscheidende Rolle /2/.
Unter diesem Blickwinkel ist der folgende Artikel zu sehen, der als Übersichtsbeitrag als Hilfestellung bei der Vorbereitung von CAD-Arbeitsplätzen dienen kann.

CAD-Software

Wesentlicher Bestandteil von CAD-Systemen ist das grafische Kernsystem (GKS). Als Programmiersprache hierfür dominiert FORTRAN. Gestützt auf das GKS lassen sich geometrische Grundsymbole (Kreise, Linien, Punkte bei 2-D-Darstellung, Körper bei 3D) zeichnen und konstruieren sowie manipulieren (Zoomen, Drehen, Spiegeln, Kopieren usw.). Daneben existieren systemspezifische Manipulations- und Konstruktionsmöglichkeiten, die sich auf komplexe Elemente beziehen bzw. das Spektrum der durch das GKS gegebenen Möglichkeiten erweitern (z.B. verschiedene Varianten zur Grundsymbolerstellung, Definition und Konstruktion von zusammengesetzten Symbolen). Zusammenfassend wird dieser die Konstruktion unterstützende Teil als Betriebssoftware bezeichnet. Der andere Teil der CAD-Software, die Anwender-Software, unterstützt den Dialog mit dem Nutzer und die Datenweiterverarbeitung entsprechend den oben genannten Punkten. Der Trend geht zu modular aufgebauten Softwarelösungen, die je nach den Forderungen des Anwenders branchenspezifisch ausbaubar sind. Ein Anwendersystem ist auf die konkreten innerbetrieblichen Bedingungen angepaßt und beherrscht die Sprache seines Fachgebietes.

Hardwarekomponenten

Ein interaktiver grafischer Arbeitsplatz weist in der Grundkonfiguration zum Beispiel den PC mit Monochromdisplay und alphanumerischer Tastatur, ein Grafikdisplay, Eingabetableau mit Lupe, Plotter und Drucker auf.

Zentraleinheit

Zur wirkungsvollen Arbeit im 2D-Bereich (z. B. Leiterplattenentwurf, Bauprojektierung, Maschinenaufstellungspläne), werden Rechner mit einer Verarbeitungsbreite von 16 Bit genutzt. Diese Rechner laufen z. B. unter den Betriebssystemen MS-DOS und SCP (CP/M). Als CPU-Schaltkreise gelangen vor allem die in Tafel 1 genannten zum Einsatz /1/. Der vom VEB Kombinat Robotron angebotene Arbeitsplatzcomputer A 7100 verwendet den K 1810 WM86 aus sowjetischer Fertigung.
Einige CAD-Systeme verwenden den 8088-Chip, der softwarekompatibel zum 8086 ist, jedoch nur über einen externen Datenbus von 8 Bit Breite verfügt. Intern wird zwar in 16-Bit-Breite gearbeitet, doch der Durchsatz ist im Vergleich zum 8086 geringer. Die Geschwindigkeit der Operationen wird durch die Implementierung von Co-Prozessoren weiter erhöht. Der verfügbare Arbeitsspeicher liegt bei mindestens 256 KByte. Im Arbeitsspeicher wird das GKS gehalten; wenn nötig wird auf extern gespeicherte Teile der CAD- und Anwendersoftware zugegriffen. Als Speichermedien werden neben Disketten zur Archivierung Festplatten mit einer Kapazität von 10... 30 MByte eingesetzt.
Will der industrielle Anwender, z. B. im Maschinenbau oder in der Fahrzeugindustrie, im 3D-Bereich konstruieren (Körper- oder Drahtmodelle mit Schattierung und Ausblendung verdeckter Körperkanten), so kommt 16- oder 32-Bit-Technik zum Einsatz. Im noch in der Entwicklung stehenden 32-Bit-Bereich werden zur Zeit vor allem die unter UNIX-Derivaten laufenden Zentraleinheiten MC 68020, NS 32032 (32332) sowie Systeme der MicroVAX-II-Architektur eingesetzt /1/. Letztere orientieren sich in Richtung der künstlichen Intelligenz (KI), der an CAD/CAM anschließenden Anwendungsrichtung der Computertechnik. KI-Systeme werden als Expertensysteme im Rahmen von CAD/CAM-Lösungen zum Einsatz kommen, z. B. zur Echtzeitdiagnose und Arbeitsplanung. Da es sich anbietet, bei KI zur Verdeutlichung logischer Strukturen und Zusammenhänge mit einer grafischen Oberfläche zu arbeiten, ist ein weiterer Berührungspunkt zu CAD (Graphics) gegeben. Ein 32-Bit-PC verfügt über einen Arbeitsspeicher von 1...16 MByte und arbeitet mit Festplatten einer Kapazität von 10... 400 MByte.
In dem 16- bzw. 32-Bit-Rechner ist eine hochauflösende Grafikkarte, teilweise ein eigenständiger Grafikprozessor, implementiert. Dieser Co-Prozessor unterstützt die Verarbeitung der Daten und geometrischen Elemente. Es kommt in vielen Fällen ein mathematischer Co-Prozessor zum Einsatz, der eine Transformierung der Modelldaten über Matrix-Gleitkommaoperationen durchführt und bei Echtzeitverarbeitung als Kinematikmodul zur Bewegungssimulation dient.
Die Grafikkarte übernimmt die Umwandlung der im Bildpunktspeicher abgelegten Informationen zu Videosignalen. Simultan darstellbar sind in der Regel 16 Farben, die teilweise aus einer größeren Farbpalette auswählbar sind und mit unterschiedlicher Intensität verwendet werden können (bewirkt räumliche Wirkung im 3D-Bereich). Es wird nicht mit dem FBAS-, sondern dem RGB-Signal gearbeitet, was eine Erhöhung der Bildqualität bewirkt. Dieser Fakt ist darauf zurückzuführen, daß das FBAS-Signal noch ein Mischsignal aus Farb-, Leuchtdichte-, Austast- und Synchronisationssignal ist, das RGB-Signal dagegen die zur Ansteuerung der Farbbildröhre nötigen Primärfarben Rot, Grün und Blau direkt liefert.
Das Bild wird ab 200 × 640 Punkten aufwärts aufgelöst. Während einem Projektanten eine Auflösung in dem Bereich bis zu 1024 × 1024 Pixeln genügt, wird auf dem Gebiet des Designs mit 2048 × 2048 Pixeln gearbeitet; Graphics, z. B. 3D-Animation, verwendet eine Auflösung bis zu 4096 × 4096 Bildpunkten.

Monitor

Zur visuellen Ausgabe wird neben dem Monochromdisplay (Text- und Datenausgabe bzw. Befehlseingabe im interaktiven Dialog) ein Farbmonitor eingesetzt, der die von der Grafikkarte bereitgestellten Videosignale verarbeitet. Die eingesetzten RGB-Monitore besitzen in der Regel eine Bildschirmdiagonale von 19 Zoll und größer. Die Auflösung der Geräte muß der Grafikkarte angepaßt sein. Ein entspiegelter Bildschirm erleichtert zusätzlich das Arbeiten. Der Trend geht zu Monitoren mit eigenständigem Grafikprozessor, so daß Grundoperationen ohne Rechnerbelastung durchgeführt werden. Dies ist besonders beim Aufbau von lokalen Netzwerken von Bedeutung, wo intelligente Arbeitsstationen durch einen übergeordneten Rechner gekoppelt sind.

Tafel 1 Gebräuchliche CPU-Schaltkreise für interaktive grafische Arbeitsplätze

CPU-Schaltkreise	Daten-/Adreßbus	Takt	adressierbarer Speicher
K 1810 WM 86 (8086)	16/20 Bit	4,77–8 MHz	1 MByte
U 8001 (Z 8001)	16/23 Bit	4 MHz	8 MByte
MC 68000	16/32 Bit	8–10 MHz	16 MByte
MC 68010	16/32 Bit	10 MHz	16 MByte
80286	16/24 Bit	6–8 MHz	16 MByte

Drucker

Der Drucker dient verschiedenen Verwendungszwecken. Zum einen ermöglicht er die Hardcopyausgabe des Grafikbildschirms. Zum anderen wird er für Stücklistenausgaben und sonstige Listings verwendet. Ausgewählt werden aus der Vielzahl der Druckerarten (z. B. Typenrad-, Thermo-, Tintenstrahl-, Laserdrucker) in den meisten Anwendungsfällen Matrixdrucker mit Einzelblatt- oder Endlospapiereinzug, da sie bei guter Ökonomie grafikfähig sind und die Möglichkeit der mehrfarbigen Darstellung bieten (z. B. robotron 1157/269). Bei einer 9×9-Matrix werden schon ansprechende Ergebnisse erzielt. Höheren Anforderungen werden Matrixdrucker gerecht, die für Graphics eingesetzt werden und bis zu 24 Nadeln besitzen. Industriellen Anforderungen genügen erstgenannte, zumal mit dem Plotter ein hochauflösendes Ausgabemedium zur Verfügung steht.

Plotter

Es wird prinzipiell zwischen zwei Arten von Plottern unterschieden. Zum einen gibt es die Flachbettplotter, die in ihrem Aufbau dem Zeichenbrett ähneln (Format AO ... A3, z. B. Digigraph). Auf dem fixierten Zeichenblatt ist der Stift in x- und y-Richtung beweglich. Anders beim Trommelplotter; hier bewegt sich der Stift nur in einer Richtung. Die Ablenkung in die zweite wird durch die Bewegung des auf eine Trommel aufgespannten Blattes realisiert, das elektrostatisch, durch Ansaugen, über Andruckrollen bzw. durch Arbeit mit Leporellopapier gehalten wird. Es wird eine Zeichengeschwindigkeit bis zu 60 cm/s erreicht. Die Begrenzung im Zeichenformat wird durch die physische Breite des Plotters gesetzt, da die Länge des Blattes keinen Forderungen unterliegt.

Plotter arbeiten mit verschiedenen Schreibstiften, wobei außer der gebräuchlichen Tusche auch Kugelschreiber oder Faserstifte eingesetzt werden. Geplottert wird meist auf Transparent- und Zeichenpapier bzw. auf Folien. Dabei wird die Kombination Zeichenmedium/Schreibstift dem jeweiligen Einsatzfall angepaßt (z. B. Leiterbahnzeichnungen auf Spezialpapier und -folien mit entsprechender Tusche). Andruckkraft des Stiftes und Stiftgeschwindigkeit sind softwareseitig beeinflußbar. Man wird im allgemeinen bei Nutzung von Transparentpapier die Stiftgeschwindigkeit herabsetzen und die Auflagekraft verringern, da ansonsten unsaubere Zeichnungen entstehen könnten. Größere Plottermodelle verwenden oft 8 Stifte bei automatischem Stiftwechsel. Die kleinste adressierbare Schrittweite liegt hier bei 0,025 mm. Eine Zeichnung ist bis auf 0,1 % genau, was für einen Menschen nicht erreichbar wäre. Da Kreise zu Polygonen angenähert werden, ist es möglich, daß diese Struktur bei großen Kreisen sichtbar wird und daß die Zeichengeschwindigkeit sinkt. Zukünftig wird auch die Intelligenz der Plotter weiter zunehmen, so daß während des Plotvorgangs die grafische Arbeit nicht unterbrochen werden muß (spezielle E/A-Schaltkreise bzw. eigene Prozessoren).

Printer-Plotter

Eine Kombination von Drucker und Plotter stellt der Printer-Plotter dar, auch Rasterplotter genannt. Er verarbeitet sowohl Grafik als auch Text, wobei die Ausgabe nach den genannten Druckprinzipien erfolgt. Da nach dem Matrix-Verfahren gearbeitet wird, werden grafische Elemente zerlegt und ihre Punktkoordinaten zwischengespeichert. Anschließend wird die aus den Daten der Einzelelemente ermittelte komplexe Matrix, die durch Sortieren der Daten in den Teilmatrizen entsteht, zeilenweise ausgegeben. Das Umwandeln und Sortieren übernimmt wiederum ein eigenständiger Prozessor. Diese Geräte arbeiten zwar schnell, besitzen jedoch eine geringere Auflösung als Trommel- oder Flachbettplotter. In der Regel wird deshalb der Verwendung einer Kombination letzterer mit einem Matrixdrucker der Vorzug gegeben.

Grafische Eingabegeräte

Neben der alphanumerischen Tastatur mit Funktionstasten dienen zur Eingabe hauptsächlich Lichtgriffel, Tableau (mit Zeichenstift und Lupe), Maus und Rollkugel, seltener Fadenkreuz und Steuerknüppel /3/. Während bei der Arbeit mit dem Lichtgriffel eine direkte Arbeit auf dem Bildschirm möglich ist (Anregen eines fotosensitiven Elementes durch den zyklischen Elektronenstrahl), arbeiten Lupe und Zeichenstift nur in Zusammenhang mit einem Digitalisiertableau. Genutzt werden induktive, kapazitive oder resistive Prinzipien. Maus und Rollkugel „merken" sich ihre Position durch Potentiometersteuerung. Aus einem Menüangebot, das entweder am Bildschirm angeboten oder auf dem Tableau dargestellt ist, kann durch Positionieren von Cursor oder Fadenkreuz die gewünschte Option im Dialog ausgewählt werden. Gezeichnet wird wiederum durch Positionierung. Im Konstruktionsmodus werden Daten eingegeben.

Programme für SCP/REDABAS

Für PC 1715 und BC A51xx wurden folgende Projekte erarbeitet:

1. Finanzbuchhaltung

- Ablösen der traditionellen Kontokarte durch Führen der Konten auf maschinenlesbaren Dateien
- programmtechnisch geführte Einrichtung und Pflege der Dateien
- beliebige unsortierte Eingabe, sortierte Verarbeitung und Ausgabe
- vollautomatische Gegenbuchung
- Rekonstruktion der Daten bei Havarie
- stets Einsicht in den Saldenstand jedes beliebigen Kontos, wahlweise Druckausgabe
- visuelle Einsicht in die Entwicklung jedes beliebigen Kontos oder Druckausgabe
- Ausgabe einer Gesamtsaldenliste, untergliedert nach Kostengruppen und -klassen
- Ausgabe der Bilanz
- automatische Übernahme der Buchungen auf Folgedisketten
- Errechnen von Durchschnittsbeständen ausgewählter Konten
- Grafikausgabe der Bestände und Durchschnittsbestände.

In allen Klein- und Mittelbetrieben anwendbar.

2. Neuererstatistik

- Das Programm ermöglicht die Datenverwaltung der Arbeitskräfte-Bestandsdatei, der Monatsrealisierung und der Neuereranalyse.
- Die Ein- und Ausgaben erfolgen über Bildschirm mit wahlweiser Druckerausgabe.
- Die Arbeitskräftebestandsdatei enthält die Planzahl der Neuerervorschläge, unterteilt nach Jugendlichen, Frauen und Produktionsarbeitern, und den wertmäßigen Plan.
- Die Neuerervorschläge und -vereinbarungen werden monatlich abgerechnet.
- In der Neuereranalyse wird die Anzahl der Neuerervorschläge und deren Nutzen kumulativ ausgewiesen.

Das Programm kann in allen Bereichen der Volkswirtschaft genutzt werden.

VEB Robotron-Vertrieb Berlin,
Werk Magdeburg,
Abt. W2/EV3, Tel. 33871/71,
PSF 136, Magdeburg, 3010

Vernetzung von Einzelsystemen

Durch den weiteren Einsatz von Multiprozessorsystemen (z. B. mit Arithmetik- bzw. Grafikprozessoren) wird die Zentraleinheit entlastet und das System leistungsfähiger. Aus der zunehmenden Ausstattung der Peripherie mit intelligenten Einheiten resultiert ein zusätzlicher Effekt. Zur Erhöhung der Auslastung von CAD-Systemen bietet sich an dieser Stelle der Aufbau von lokalen Netzwerken an. Dabei werden in der ersten Ausbaustufe periphere Einrichtungen wie Plotter und Massenspeicher durch mehrere Zentraleinheiten gemeinsam genutzt, was zu einer spürbaren Verbesserung der Ökonomie führt. In der zweiten Stufe wird zur Bildung von Mehrplatzsystemen übergegangen. Hier werden intelligente Arbeitsstationen verwendet, die eine Vielzahl von Operationen eigenständig durchführen und bei Bedarf auf einen Host-Rechner zugreifen können (Arbeit mit Mitteln der dezentralen Intelligenz). Da für CAD-Systeme auf 16-Bit-CPU-Basis eine Rechnerauslastung von 90 % eingeschätzt wird /4/, ist als Zentralrechner ein 32-Bit-PC eingesetzt. Anderenfalls entstehen recht lange Warte- bzw. Antwortzeiten.

Zum Zeitpunkt des Verfassens lag noch kein Anwendungsfall zum 32-Bit-Prozessor 80386 vor, die sich jetzt häufen. Deshalb wird davon ausgegangen, daß künftig insbesonders 3D-Systeme mit dieser CPU und den ebenfalls entwickelten Coprozessoren arbeiten werden. Als mögliche Betriebssysteme kommen ein UNIX-Derivat oder MS-DOS V5.0 bzw. andere aus Version 3.2 entwikkelte Varianten zum Einsatz.

Literatur

/1/ Hellwig, H.-E.: Neue CAD-Systemarchitekturen. VDI-Zeitung 127 (1985) 14, S. 563
/2/ Haas; Hentschel; Lüders: Erfahrungen bei der Einführung eines CAD-Arbeitsplatzes. Textiltechnik 36 (1986) 9, S. 466
/3/ Duus, W.: CAD-Systeme. Springer, 1983
/4/ Koch, J.: Vernetzte PC oder Mehrplatzsysteme? IC-Wissen 1986 9/10, S. 8–10

⊠ KONTAKT ☎

VEB Zentrales Projektierungsbüro der Textilindustrie, Dr.-Kurt-Fischer-Str. 31, Leipzig, 7010; Tel. 20 95 41



In Leipzig fand vom 18. bis 22. Mai dieses Jahres die 21. Angebotsmesse „Wissenschaftlich-technische Ergebnisse" statt, auf der aus den Industriebereichen Elektrotechnik und Elektronik, Allgemeiner Maschinen-, Landmaschinen- und Fahrzeugbau sowie Schwermaschinen- und Anlagenbau nachnutzbare Beispiellösungen gezeigt wurden. Vor allem das umfangreiche Angebot von Hard- und Softwareentwicklungen dürfte zum Erfolg der Messe – die einen neuen Besucherrekord verzeichnete – beigetragen haben. Auch hier zeigte sich, wie notwendig und gefragt solche Veranstaltungen sind.
Um einem noch breiteren Anwenderkreis Unterstützung bei der Arbeit mit dem Computer zu geben, stellen wir in unserer Rubrik Börse einige der gezeigten Exponate vor.

EMR-Prozeßsteuerung

Besonders als Steuerkern für Rationalisierungsmittel geeignet ist dieser in Zusammenarbeit mit der TU Dresden entwickelte Einkartenrechner mit 24 Ein- und Ausgängen. Es können wahlweise Einchipmikrorechner UB 8820 oder UB 8840 eingesetzt werden. Beim UB 8820 stehen dem Anwender bis zu 4 KByte EPROM und beim UB 8840 bis zu 6 KByte zur Verfügung. Die Leiterkarte hat einen zusätzlichen RAM von 2 KByte und ist zu den Leiterkarten des K-1520-Systems kompatibel.
Die Taktfrequenz kann zwischen 4 MHz (bei 8-MHz-Schwingquarzen) und 2,5 MHz (bei 10-MHz-Schwingquarzen) gewählt werden.
VEB IFA-Karosseriewerke Dresden, Sitz Radeberg, PSF 5, Radeberg, 8142.

Emulatormodul

Der für den KC 85/1 geschaffene Modul hat folgende Charakteristik:
– Entwicklung von EMR-Programmen, wobei die schrittweise Simulation des entwickelten Programms möglich ist; Echtzeitlauf des EMR-Programms über eingebauten EMR.
Alle EMR-Ports sind über Steckverbinder nach außen geführt, somit kann die Funktionsfähigkeit mittels externer Einrichtungen überprüft werden.
– Mehrrechnersystem; EMR kann abgeschlossene Teilaufgaben im Gesamtkonzept übernehmen (Master-Slave-Prinzip). Korrespondenz zwischen EMR- und Masterrechner (U880) erfolgt über PIO-Port.
– Es ist eine schnellere Entwicklung von EMR-Programmen auf Wirts-Rechnern möglich, da Programmierfehler durch Echtzeitbetrieb sofort korrigiert werden können.
VEB Meßgerätewerk „Erich Weinert" Magdeburg, Straße der DSF, Magdeburg, 3011

CMOS-Speicher für PS 2000

Die Baugruppe ist eine 2-Ebenen-Leiterplatte, welche ohne zusätzliche Änderungen im Programmiergerät PRG 2000 der freiprogrammierbaren Steuerung PS 2000 mit der vorhandenen RAM-Karte ausgetauscht werden kann.
Der Einsatz dieser CMOS-Karte ermöglicht das Ablegen eines kompletten PS-2000-Programmes (Speicherbedarf 4 KByte – alte Karte nur 1 KByte). Weiterhin bleibt das eingegebene Programm bei Netzausfall erhalten, da die CMOS-RAMs mit Akkus gepuffert werden.
VEB dkk Scharfenstein, BT Eleba, Scharfenstein, 9366

KC 85/3 als intelligente Bedieneinheit

Kernstück des Exponates ist eine Koppellogik, die es ermöglicht, einen KC 85/3 zur Fehlersuche an K1520-Mikrorechner-Systemen einzusetzen. Die Bedieneinheit realisiert in ihren Betriebsarten folgende prinzipielle Funktionen:
– Setzen von Haltepunkten (triggerbar auf Adressen und/oder auf Steuersignale und/oder deren Kombinationen)
– Abarbeitung von Programmen im Schrittbetrieb ab Haltepunkt (mit variabler Schrittweite) mit Refresh-Erzeugung
– Anzeige von Daten- und Adreßbus auf Bildschirm, Steuerbus über LED
– direkter Speicherzugriff zur zu prüfenden MR-Einheit (DMA mit Bildschirmanzeige)
– Manipulationsmöglichkeit der Abarbeitungsfolge von Anwenderprogrammen (auch ROM-residenter Software)
– Einfügen von Serverroutinen an beliebigen Triggerpunkten und deren Abarbeitung im Schritt- oder Echtzeitbetrieb.
VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, Eisenacher Str. 40, Mühlhausen, 5700

Prüflingsaufnahme zur Anwendung der Signaturanalyse

Die Prüflingsaufnahme (PLA) K1520 dient der Fehlersuche auf bestückten Leiterplatten (BLP) K1520, die speziell in den Geräten A 5120/A 5130 und davon abgeleiteten Gerätevarianten zum Einsatz kommen.
Mit Hilfe eines BC A 5120/A 5130, eines Signaturanalysators robotron 31020 und handelsüblicher Meß- und Prüfmittel, wie Service-Oszillograf und Digital-Multimeter, läßt sich in Verbindung mit der PLA K1520 ein Prüfarbeitsplatz für BLP des K1520-Systems konfigurieren. Dabei ist die PLA K1520 über Kabel mit dem Bus des Bürocomputers verbunden. Die zu prüfende BLP wird in die PLA K1520 gesteckt und durch softwaremäßige Nachbildung der Bus-Signale des BC stimuliert. Durch programmgesteuerte Bedienerführung ist es möglich, alle Anweisungen zur Bedienung des Systems und alle Meßpunkte mit dazugehörigen Signaturen über den Bildschirm auszugeben.
VEB Robotron Buchungsmaschinenwerk Karl-Marx-Stadt, Annaberger Str. 93, Karl-Marx-Stadt, 9048; Tel. 5716268 (Koll. Schmidt)

Akustikkoppler

Das Kopplungsgerät dient zur Verbindung zweier Rechner, indem die per Telefonleitung zu übertragenden Daten aufbereitet und umgesetzt werden. Die Empfangs- und Sendedaten werden dabei über eine V.24-Schnittstelle akustisch an das Telefon gekoppelt. Ein Eingriff in das Telefon erfolgt nicht! Die Verwendung ist für Datenübertragung bis 200 Baud vorgesehen.
VEB Weimar-Werk, Buttelstedter Straße 4, Weimar, 5300

Universelles EPROM-Löschgerät

Eine höhere Störsicherheit – da keine IC-Fassungen nötig sind – und eine schnelle Programmkorrektur ergeben sich mit diesem Löschgerät.
Charakteristik:
· Löschung von EPROM-IC aller Typen sowie Leiterkarten bis 130 x 200 mm mittels UV-Lampe
· Löschzeit vorwählbar 10 ... 100 min
· Sicherheitsabschaltung der UV-Lampe bei Öffnung des Gerätes
· Betriebsanzeige durch 2 LED
· 220 V
· Abmessungen: 23 x 18 x 15,5 cm^3
VEB Grubenlampen- und Akkumulatorenwerke Zwickau, Reichenbacher Str. 89, Zwickau, 9502

Logikanalysator

Die Lösung stellt mit 32 Kanälen / 20 MHz, leistungsfähiger Triggerlogik, Erweiterungsbaugruppen und der möglichen Teilnutzung auch als Bürocomputer einen komfortablen Logikanalysator dar.
Die Einstellung erfolgt softwaregesteuert bzw. über Menü von Diskette.
Ein für die Verwaltung von dezentralen Lagern erarbeitetes Materialprojekt kann gleichzeitig zur Stückleistenherstellung genutzt werden.
Als Hardwarebasis dienen ein BC A 5120 und der von der Technischen Universität Dresden entwickelte LA 32/20.
VEB Robotron-Elektronik Riesa, Pausitzer Str. 60, Riesa, 8400

Softwareanalysegerät

Das autonome Softwareanalysegerät ermöglicht die Fehlersuche in U-880-Assemblerprogrammen im Echtzeitlauf. Es ermöglicht die Optimierung von Regelalgorithmen. Aus dem laufenden Programm werden ohne zeitliche Beeinträchtigungen Datenproben in einen speziellen Debuggingspeicher eingeschrieben. Durch Markierung charakteristischer Programmteile und die Registrierung dieser Marken während des Programmablaufs ist eine Rekonstruktion des Programmverlaufs in stark verzweigten Programmen möglich. Eine Analogendausgabe von Datenfolgen ist im Echtzeit- und Debuggingspeicher-Wiedergabebetrieb möglich.
VEB Mikroelektronik „Karl Liebknecht", Ruhlsdorfer Weg, Stahnsdorf, 1533

Komplexprogramm technologisch-ökonomische Kennziffern

Das Programm ist eine BC/PC-Lösung auf der Basis des Betriebssystems SCP und REDABAS. Es wertet Kosten-, Material- und Aufwandsinformationen aus dem betrieblichen Abrechnungsprozeß (z. B. BAB II) zum Zwecke zielgerichteter Analysentätigkeit im Sinne der Schwerpunktfindung für technologisch-ökonomische Maßnahmen aus.
Dabei werden Aussagen/Analysen für Kostenträger u./o. Erzeugnisgruppen sowohl für einzelne Betriebe als auch für das Kombinat hinsichtlich
– material-ökonomischer Kennziffern
– Arbeitszeitaufwandskennziffern
– weiterer ökonomischer Kennziffern
ermittelt und dargestellt.
Kombinat VEB Carl Zeiss JENA, Fachdirektion Technologie und Rationalisierung, Carl-Zeiss-Str. 1, Jena 6900

CAM-Bildschirmarbeitsplatz für Meister und Disponenten

Der Arbeitsplatz findet Anwendung als Dialogarbeitsplatz für Meister und Disponenten der Teilefertigung.
Am Bildschirmarbeitsplatz (BC A 5130) können die Disponenten Produktionsvorgaben für ihren Bereich abrufen und im Dialog die Maschinenbelegung vornehmen. Die Meister des Bereiches können sich die Maschinenbelegung ausgeben lassen und melden den Arbeitsfortschritt zurück.
Der Arbeitsplatz ist über Datenfernübertragung mit dem Produktionssteuerrechner A 6402 verbunden.
Kombinat VEB Carl Zeiss JENA, Betrieb Gera, Keplerstraße 100, Gera, 6500

Berechnung der Fertigungszeitentwicklung

Mit dem entwickelten Programm in BASIC, zugeschnitten für den KC 85/2, wird der benötigte Fertigungszeitaufwand pro Erzeugnis und Fortschrittszahl sowie der durchschnittliche Fertigungsaufwand über einen definierten Fertigungszeitraum des Serienanlaufes berechnet und ausgewiesen.
Die Berechnung der Fertigungszeit neuer Erzeugnisse und damit der Fertigungszeitentwicklung erfolgt nach einer den Gesetzmäßigkeiten der Großserienfertigung abgeleiteten mathematischen Beziehung.
VEB IFA-Motorenwerke Nordhausen, Freiherr-v.-Stein-Straße 30c, Nordhausen, 5500

CAD/CAM-System für die konstruktiv-technologische Vorbereitung und Fertigung im Automatisierungs- und Anlagenbau

Das CAD/CAM-System dient der Rationalisierung der konstruktiv-technologischen Vorbereitung und der Fertigung auf Basis des Einheitlichen Gefäßsystems EGS und des Rahmenaufbau-Systems RAS.
Das rechnergestützte Verfahren umfaßt folgende Vorgänge:
1. Erfassen von Schaltteillisten (Entwurf)
2. Entwerfen der Anlagen/Gerät-Belegung
3. Zeichnen von Belegungsplänen
4. Erzeugen von NC-Daten
5. Erstellen von Stücklisten (SL)
6. Übertragung von SL-Daten an zentrales betriebl. Anwendungssystem (SAPKO)
7. Erstellen von Zeitaufrechnungsbelegen (technologische Arbeitsgangbeschreibung) und
8. Pflege der dezentralen Datenbasis.

Zur rechentechnischen Basis gehören: SKR-Rechneranlage, graf. Bildschirmgerät (RSG K 8917 o. ä.), Zeichengerät (DZT 90 x 120 o. ä.), Betriebssystem OS-RW V 2.8, Datenverwaltungssystem DTR, Graphik-Programmpaket VEMGKS.
Es sind auch Teillösungen des durchgängigen CAD/CAM-Systems nachnutzbar; bei Nachnutzung der CAM-Lösung (Vorg. 3) muß zusätzlich die Lösung der NVe (Nr. 607/5/3) „NC-Lochband-Steuerung für RAS-Fügehalbautomat" angewendet werden.
VEB Elektroprojekt und Anlagenbau Berlin, Rhinstr. 100, Berlin, 1140

Schaltplanerstellung am A 5120/A 5130

Mit Hilfe des angebotenen Programmpaketes können schematische Zeichnungen unterschiedlicher Art (Funktionsschalt-, Logik-, Stromlaufpläne u. dgl.) am Bildschirm im Dialog erstellt, auf Diskette gespeichert und über Drucker ausgegeben werden.
Basis der quasigrafischen Darstellungen bildet ein maximal 128 Zeichen betragender Halbgrafikzeichensatz, der auf EPROM gespeichert ist. Mittels zugehöriger Generierprogramme ist dieser Zeichensatz vom Anwender entsprechend seinen speziellen Anforderungen modifizierbar. Zum effektiven Bildaufbau kann der Anwender höher integrierte zeichnerische Darstellungen wie Zeichenketten, Teilbilder (maximal Bildschirmgröße) und komplette Bilder erstellen und in Stammdateien zur Verwendung ablegen.
Die Zeichnungen sind in den drei Formaten A4-hoch, A3-quer oder A3-hoch zu erstellen.
Voraussetzung der quasigrafischen Arbeitsweise ist die Modifikation der Bildschirmsteuerung durch Einsatz der ABS K 7029,00.
Das Programmsystem arbeitet im Betriebssystem UDOS 1526, Vers. 4.0.
Rationalisierungseffekte werden beim Einsatz des Programmsystems, insbesondere durch Wiederverwendung von Zeichnungen bei geringer Modifizierung der Darstellung erzielt.
Technische Daten:
– BC A 5120/30 mit Bildschirmkarte ABS K 7029,00
– Drucker SD 1154/5500 oder SD 1157/269
– Betriebssystem UDOS 1526, Version 4.0
VEB Elektroprojekt und Anlagenbau Berlin, Rhinstr. 100, Berlin, 1140

Absatzprojekt

Dieses Projekt dient besonders der Rationalisierung der Fakturierung. Das Projekt Absatz besteht aus mehreren autonomen Teilkomplexen, aus welchen sich ein spezielles Anwendersystem installieren läßt.
Durch Verwendung des Betriebssystems SCPX 1526, der Programmiersprache BASIC und des relationalen Datenbanksystems REDABAS kann das Projekt auch auf PC 1715 oder AC A 7100 eingerichtet werden.
Teilkomplexe:
– Abrechnung Export
– Aufbau und Druck von Marktrechnungen, Währungsfaktura und Lieferspezifikation
– Abrechnung Inland Lieferscheine/Rechnung
– Abrechnung Ersatzteile Aufbau und Druck der Ausgangsrechnung
– Abrechnung Lohnaufträge
– Umsatzstatistik
– Dateiänderungsdienst
VEB Sachsenring Automobilwerke Zwickau, Abt. TN, PSF 311-313, Zwickau, 9541

Auftragsplanung und -steuerung

BS-APS ist ein durchgängig modulares System für die Belange der innerbetrieblichen Auftragsabwicklung, von der Auftragsannahme bis zum Versand. Es baut auf einem umfangreichen Schnittstellenkonzept auf und ist von autonomen Arbeitsplätzen (APS-A) bis zu Verbundsystemen (APS-V) stufenweise ausbaufähig.
BS-APS ist vorgesehen für folgende Konfigurationen:
1. SKR (K 1630, A 6422), A 7100
2. BC/PC + KC (als grafikfähiges u. farbtüchtiges UBT)
3. KC
4. APS-V (lokales Rechnernetz mit Konfiguration von 1. u. 2.)
VEB Forschung, Entwicklung und Rationalisierung des Schwermaschinen- und Anlagenbaus, Bleckenburgstr. 25, Magdeburg, 3011

Textverarbeitung

Mit dem Projekt Textverarbeitung wird eine wesentliche Rationalisierung der Verwaltungsarbeit im Bereich ORZ erreicht. Es gliedert sich in folgende Teilgebiete:

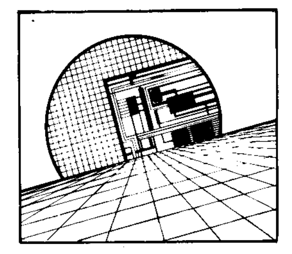
Leitungsorganisationsarbeit umfaßt im wesentlichen die BC-gestützte Bearbeitung der Strukturkurzzeichen und Telefonanschlüsse, Organisationshandbuch-Anweisungen, Ordnungen des übergeordneten Organs, Verzeichnisse der Betriebsobjekte, Vordrucke, ausgewählten Anweisungen sowie des Informationsspeichers zur Verteilung von Büro- und Vervielfältigungstechnik.

Terminkontrolle beinhaltet die Übersicht und Kontrolle von Terminen zum übergeordneten Organ, zum Leiter und innerhalb der Struktureinheiten.

Ablösung der Adrema-Technik mittels Etikettiereinrichtung wird vorrangig für Druck der Arbeitszeitnachweise, Banderolen, Anwesenheitslisten, Inventurnachweise und weiterer Formblätter verwendet.

Texthandbuch beinhaltet Standardbriefe, Standardschreiben in Form von Textkonserven.

VEB Transformatorenwerk „Karl Liebknecht", Wilhelminenhofstr. 83–85, Berlin, 1160



Übersicht 1

Gate-Array-System U5200

Was bietet der VEB ZMD hierzu an?
- Master
- Makrozellenkatalog mit etwa 100 Strukturen
- Anwenderberatung
- durchgängiges Entwurfssystem
- Entwurfsbetreuung
- Lieferung von Mustern im 64poligen Plast-Chip-Carrier-Gehäuse (PCC64)
- Produktion

*Was ist anwenderseitig **erforderlich**?*
- Umsetzung der Schaltung in Gate-Array-Logik
- Logik- und Dynamik-Simulation
- Erprobung der anwenderspezifischen Musterschaltkreise

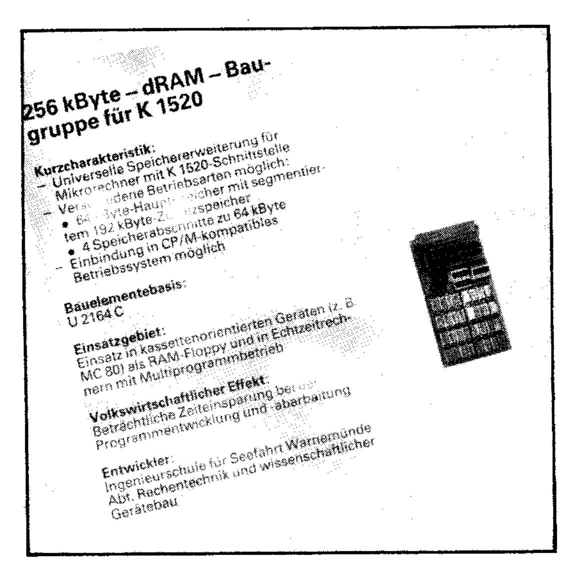
Technisch-organisatorischer Ablauf
- Kundenberatung
- Vertrag
- Gate-Array-Layout
- Freigabe des Entwurfes durch den Anwender
- Musterfertigung
- Erprobung der Muster im Einsatzfall
- Produktionsfreigabe durch Anwender
- Produktion

Standardzellen-System U1500/U1520

Was bietet der VEB ZMD hierzu an?
- Standardzellenkatalog mit ca. 40 Strukturen
- Anwenderberatung
- durchgängiges Entwurfssystem
- Entwurfsbetreuung
- Lieferung von Mustern mit standardisierten PIN-Zahlen in entsprechendem Gehäuse
- Produktion nach Bilanzentscheid im VEB ZMD

*Was ist anwenderseitig **erforderlich**?*
- Schaltungsentwicklung
- Erfassung der Schaltung in Netzbeschreibungssprache (NBS)
- Textfolgeerstellung
- Logik- und Dynamiksimulation mit dem Programm „KOSIM“
- Layoutgenerierung mit dem Programm „STAMA“
- Funktionstext der Musterschaltkreise

Technisch-organisatorische Arbeit
- Kundenberatung
- Schaltungsentwicklung
- Vertrag über Musterpräparation
- Schaltungserfassung und Testfolgenerstellung
- Absichtssimulation (Logik-Simulator)
- Plazierung/Trassierung (Layoutgenerierung)
- Dynamik-Simulation
- Verifikation
- Testpatterngenerierung entsprechend Testerspezifikation
- Freigabe des Layouts durch Anwender
- Musterpräparation
- Scheibenmessung
- Musterverkappung
- Erprobung der Muster im Einsatzfall
- Produktionsfreigabe durch Anwender
- Produktion



256 kByte – dRAM – Baugruppe für K 1520

Kurzcharakteristik:
– Universelle Speichererweiterung für Mikrorechner mit K 1520-Schnittstelle
– Vers[illegible]dene Betriebsarten möglich:
• 64 [illegible]yte-Haupt[illegible]speicher mit segmentiertem 192 kByte-Z[illegible]speicher
• 4 Speicherabschnitte zu 64 kByte
– Einbindung in CP/M-kompatibles Betriebssystem möglich

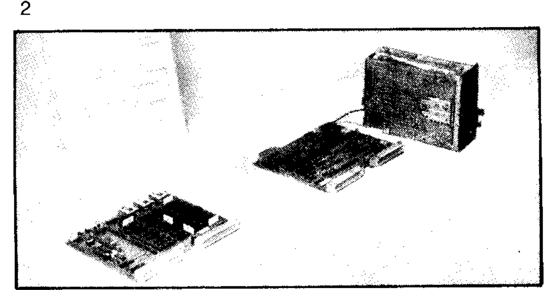
Bauelementebasis:
U 2164 C

Einsatzgebiet:
Einsatz in kassettenorientierten Geräten (z. B. MC 80) als RAM-Floppy und in Echtzeitrechnern mit Multiprogrammbetrieb

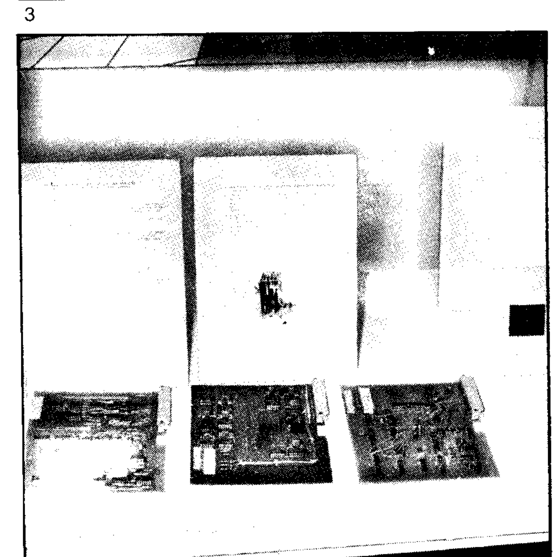
Volkswirtschaftlicher Effekt:
Beträchtliche Zeiteinsparung bei der Programmentwicklung und -abarbeitung

Entwickler:
Ingenieurschule für Seefahrt Warnemünde Abt. Rechentechnik und wissenschaftlicher Gerätebau

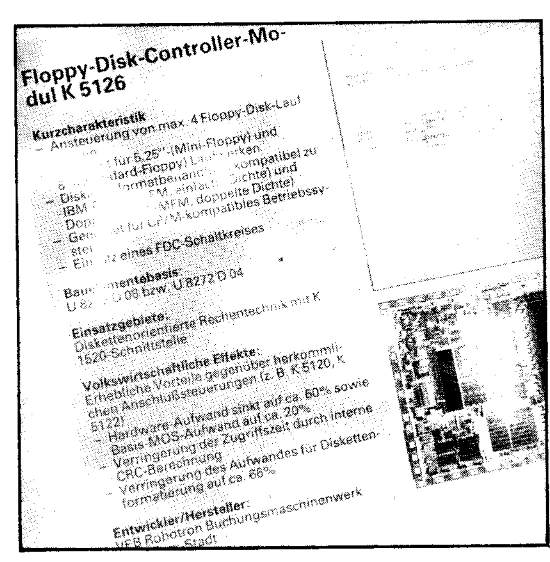
2



3



5



Floppy-Disk-Controller-Modul K 5126

Kurzcharakteristik
– Ansteuerung von max. 4 Floppy-Disk-Lauf[illegible] für 5,25"-(Mini-Floppy) und [illegible]dard-Floppy) La[illegible]rken
[illegible] kompatibel zu [illegible]ormatbe[illegible]ichte) und
– Disk[illegible] FM, einfa[illegible] Dichte) und
IBM [illegible] MFM, doppelte Dichte)
Dop[illegible] für CP/M-kompatibles Betriebssy-
– Ge[illegible]
ste[illegible]
– Ein[illegible]z eines FDC-Schaltkreises

Bau[illegible]mentebasis:
U 8[illegible] D 08 bzw. U 8272 D 04

Einsatzgebiete:
Diskettenorientierte Rechentechnik mit K 1520-Schnittstelle

Volkswirtschaftliche Effekte:
Erhebliche Vorteile gegenüber herkömmlichen Anschlußsteuerungen (z. B. K 5120, K 5122)
– Hardware-Aufwand sinkt auf ca. 60% sowie Basis-MOS-Aufwand auf ca. 20%
– Verringerung der Zugriffszeit durch interne CRC-Berechnung
– Verringerung des Aufwandes für Diskettenformatierung auf ca. 66%

Entwickler/Hersteller:
VEB Robotron Buchungsmaschinenwerk [illegible] Stadt

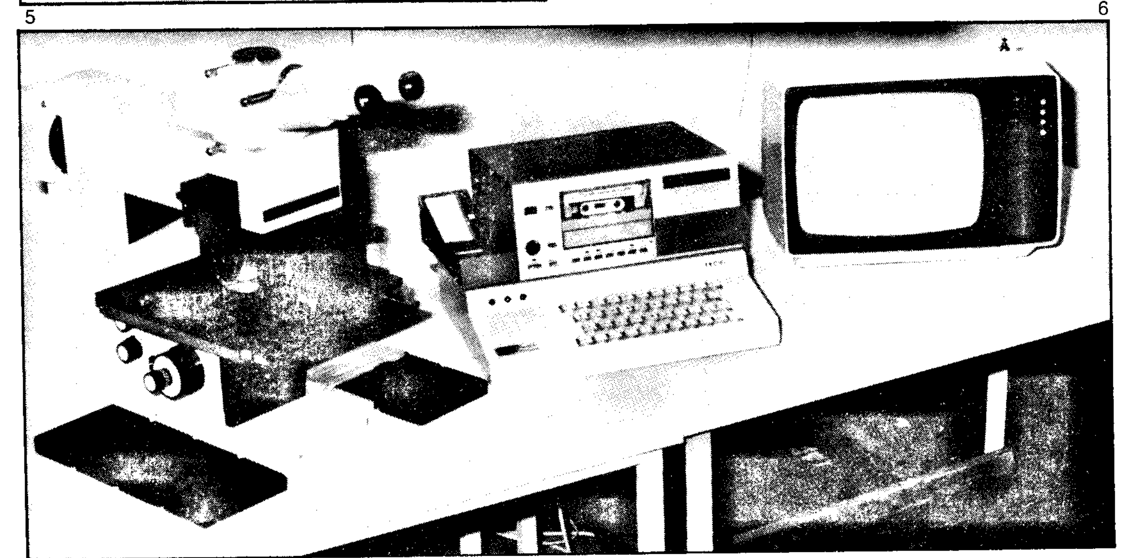
4

Übersicht 2

Gegenüberstellung **Gate-Array-System** U5200 mit Standardzellen-System U1500/U1520

	Gate-Array-System	Standardzellensystem
Integrationsgrad	12000 Transistoren	max. 13000 Transistoren (bei 7,5 × 7,5 mm², CSGT2s)
Anzahl der möglichen, verschiedenen Funktionselemente	ca. 100	ca. 40
Layout	für jeweiligen Master festgelegt	kundenspezifisch
Technologie	CSGT2s	CSGT2n (U1500) CSGT2s (U1520)
Gatterlaufzeit	5 ns	5 ns
(typ.) Stückzahl	max. 10000 Stück/Jahr	10000 ... 100000 Stück/Jahr
Gehäuseart/Zahl der Anschlüsse	PCC 64	DIP 16, DIP 18, DIP 24, DIP 28, DIP 40, PCC 64, QFB 68
Betriebsspannung	4,75 ... 5,25 V	4,75 ... 5,25 V
Takt	abhängig von kundenspez. Logik, typ. 1 ... 4 MHz	abhängig von kundenspezifischer Logik, typ. 4 MHz
TTL-Kompatibilität	ja	ja
Aufwandverhältnis bzgl. Entwurfszeit und Kosten	1 : 2	

6



ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 8
S. 225–256

1

CAD/CAM-Systeme

Die CAD/CAM-Technologie gewinnt für alle Bereiche unserer Volkswirtschaft zunehmend an Bedeutung. Vom VEB Kombinat Robotron wird für CAD/CAM-Aufgaben ein umfangreiches Sortiment an Rechnern, Peripherie und Basissoftware angeboten, das an die verschiedenartigsten Anwenderbedürfnisse angepaßt werden kann. Bild 1 zeigt die Neuheiten des Kombinates zur Leipziger Frühjahrsmesse 1987 zu dieser Thematik auf einen Blick.

Als unterste Stufe eines CAD/CAM-Systems können Bürocomputer A5120/30 und Personalcomputer 1715 (Bild 2) mit entsprechender Peripherie genutzt werden. Schon leistungsfähiger – und damit für anspruchsvollere Anwendungen geeignet – ist der 16-Bit-Arbeitsplatzcomputer robotron A7100. Ebenfalls für den Einsatz in CAD/CAM-Systemen ist das Interaktive grafische Terminal robotron K8918 (Bild 3) vorgesehen. Es wird für Dialog-Arbeitsplätze genutzt, die unter Steuerung eines Hostcomputer operieren und mit Grafischer Kernsystem-Software (GKS) unterstützt werden. Als Hostcomputer kommen außer dem Mikrorechner K1630 und weiteren SKR-Rechnern ESER-Rechner mit höherer Verarbeitungsleistung zur Anwendung.

Der Beitrag *Komponenten von CAD-Arbeitsplätzen* von Christian Haas im gleichen Heft geht ausführlicher auf den Aufbau solcher Systeme ein.


2


3

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

Grund

MP Mikroprozessortechnik

Heft 9 · 1987
VEB Verlag Technik Berlin
ISSN 0232-2892



C571
C570

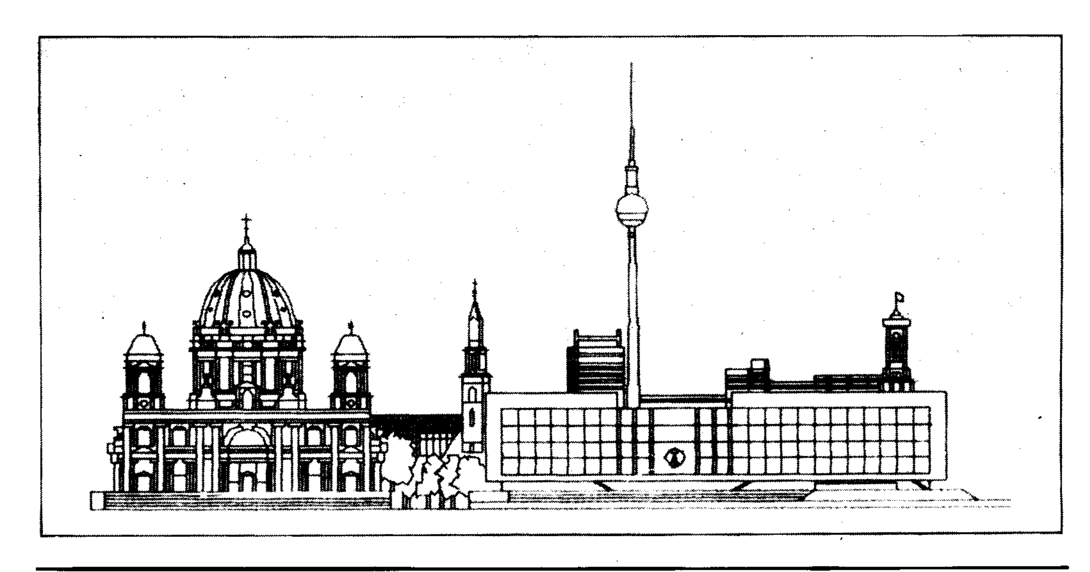
○ **Schwerpunkt: A/D-Wandler**

○ **MP-Kurs: PASCAL**

Neu: MP-Computerclub

Die Ausstellung „Wissenschaft und Produktion der DDR im Dienste des Volkes“ anläßlich des 750jährigen Bestehens von Berlin hat ihre Pforten geschlossen.
Anhand etwa 1300 Exponaten – die zum großen Teil in Funktion gezeigt wurden – konnten sich die Besucher von der Leistungskraft unserer Volkswirtschaft überzeugen.
In dem großen Komplex Mikroelektronik dominierten im Eingangsbereich die Grafik-Terminals, die mittels Datenfernübertragung an den neuen 32-Bit-Computer von Robotron angeschlossen waren (Bild 1). Gemeinsam mit den modernen Anlagen des VEB Carl Zeiss JENA – z. B. dem automatischen Überdeckungsrepeater zur Belichtung von Wafern – bildet diese Technik die Voraussetzung zur Entwicklung und Produktion hochintegrierter Schaltkreise.
Für ein breites Einsatzspektrum vorgesehen ist der 16-Bit-Computer A 7150, der mit seinen Verbesserungen gegenüber dem A 7100 vor allem CAD-Aufgaben auf höherem Anforderungsniveau bearbeiten kann (Bild 2). Mit dem neuen Betriebssystem DCP 3.1. wird sowohl die Brücke zu der weit verbreiteten MS-DOS-kompatiblen Standardsoftware geschlagen als auch Programmkompatibilität zu künftigen 16-Bit-Personalcomputern hergestellt.
Umfangreiche Aufgabenlösungen wurden mit der mehrplatzfähigen CAD/CAM-Arbeitsstation auf der Basis des K-1600-Systems demonstriert (Bild 3).
Inzwischen weit verbreitet sind für Aufgaben in den unterschiedlichsten Bereichen unserer Volkswirtschaft 8-Bit-Computer wie der PC 1715, dessen Weiterentwicklung 1715W mit dem vergrößerten Hauptspeicher und dem modifizierten Betriebssystem eine komfortablere und leistungsfähigere Benutzeroberfläche gewährleistet. Noch besser ausschöpfen kann man die Ressourcen der Personalcomputer oft, wenn sie nicht als Einplatzsystem, sondern im Verbund arbeiten. Eine praktische Demonstration dafür gab die Humboldt-Universität zu Berlin mit dem lokalen Netz ROLANET. Es nutzt als Übertragungsmedium Lichtwellenleiter, die über passive optische Sternkoppler und separate Knotenrechner die PC miteinander verbinden (Bild 4). Somit kann zum Beispiel teure Peripherie von allen angeschlossenen PC gemeinsam genutzt werden.

(Fortsetzung siehe 3. Umschlagseite)



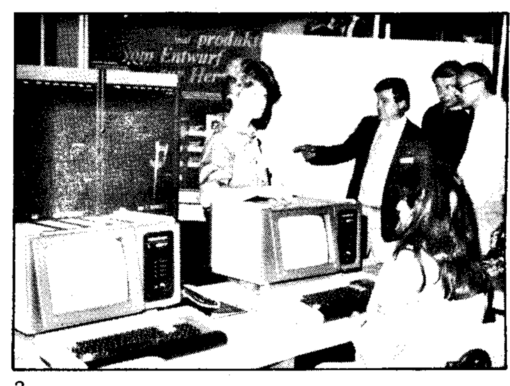
Leistungsfähige Computertechnik in allen volkswirtschaftlichen Bereichen



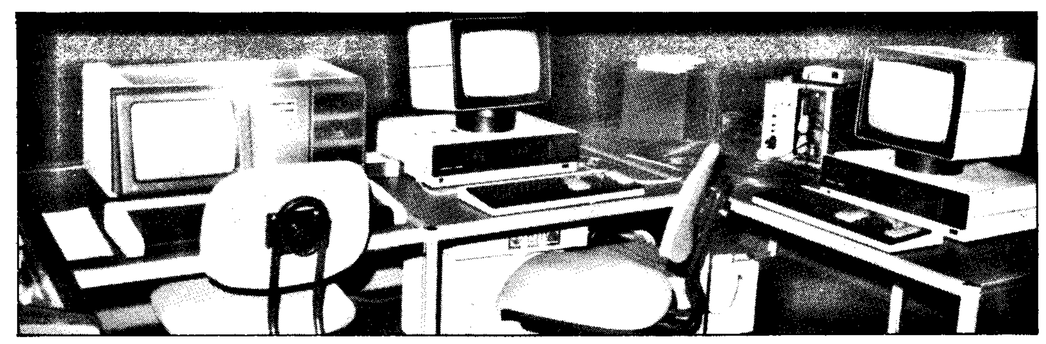
1



2

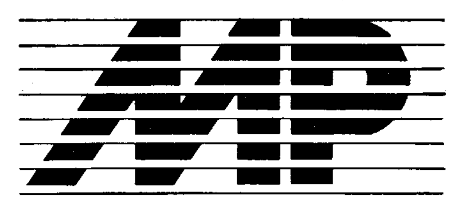


3



4

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 9 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 2 87 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2 87 02 03); Hans Weiß, Redakteur (Tel.: 2 87 03 71); Sekretariat Tel.: 2 87 03 81

Gestaltung Christina Kaminski (Tel.: 2 87 02 88)

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

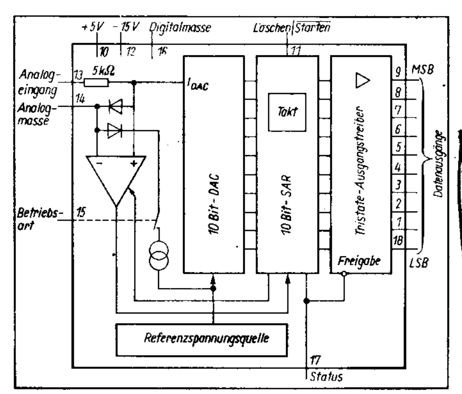
Redaktionsschluß: 24. Juli 1987

AN (EDV) 49837

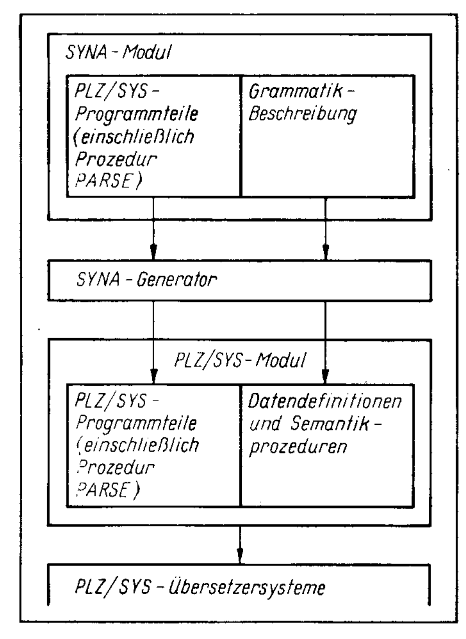
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010, und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Seite 260



Seite 275

Zum Titelbild

Ein Spitzenexponat auf der Ausstellung „Wissenschaft und Produktion der DDR im Dienste des Volkes" war die von der Friedrich-Schiller-Universität Jena entwickelte und im VEB Mikrofontechnik Gefell produzierte automatische Laser-Beschriftungsanlage ALBA. Auf ihr entstand das im Titelbild gezeigte Ausstellungssignet als Gravur in eine Plasttafel. Die Beschriftung von Plastschildern für die Industrie ist zwar ein Haupteinsatzgebiet der Anlage, jedoch erlaubt ihre Vielseitigkeit auch die Gravur bzw. Strukturierung von Metallen, Hartmetallen, Gummi, Keramik und Glas.
Der Strahl des cw-Neodym-YAG-Lasers wird dabei von einem Mikrorechner – z. B. MC 80 – gesteuert und über die verfahrbare Optik auf das Material fokussiert.

Inhalt



Dialog

✉ Veröffentlichung von Programmen im DUMP-Format

Die Veröffentlichung von Softwarelösungen als Anregung und Vorbild hat zweifellos eine große Bedeutung für die Qualifikation und in eingeschränktem Maße auch für die Nachnutzung. Es ist erfreulich, daß mit Ihrer Zeitschrift jetzt das passende Podium dafür existiert.
Setzt man eine qualifizierte Kommentierung voraus, gibt es bei der Veröffentlichung von Quelltexten auch keine Schwierigkeiten. Ein Ausweichen auf das DUMP-Format macht das Unterfangen fragwürdig (s. Schlenzig; Heft 1; 3. US und Lennartz; Heft 4; S. 126).
Sofern nicht Platzgründe dazu zwingen, ist die Veröffentlichung der Assembler-Quelltexte zu fordern. Nichts ist frustrierender, als in fremden Programmen die Fehlerursache zu suchen, wenn nur der Maschinencode vorliegt. Dieser Beschäftigung fröne ich trotz Kontrollsummen beim Artikel von Schlenzig schon eine Weile.
Ist die Veröffentlichung im DUMP-Format unumgänglich, müssen Datensicherungsverfahren, die über Prüfsummen hinausgehen, eingesetzt werden. Ein entsprechendes Verfahren wäre in MP zu veröffentlichen. Ein Beispiel für eine derartige Routine ist von Heyder im Funkamateur 11/86, Seite 565 beschrieben. Druckfehler (in beiden Artikeln finden sich Beispiele) machen die Wiedergabe als Faksimile eines Computer-Ausdrucks wünschenswert.
Die Prüfung der Software ist von Ihrer Seite aus sicher schwierig, erscheint aber speziell bei DUMP-Veröffentlichungen im Interesse der Leser für sinnvoll.

Dr. Volkmar Richter, Paretz

Sie sprechen ein Problem an, daß uns seit längerem beschäftigt.
Unabhängig davon, ob es sich um Quelltexte oder HEX-DUMPs handelt, streben wir an, die Programme original von der Vorlage, die wir vom Autor erhalten, zu veröffentlichen (Faksimile). D. h., sie werden für den Druck in der Zeitschrift durch die Druckerei vom Programmausdruck reproduziert. Dabei ist es besonders wichtig, daß weißes Papier und ein möglichst neues, schwarzes Farbband verwendet werden. Nur mit dieser Verfahrensweise lassen sich Fehler, die beim Satz des Programmtextes entstehen könnten, vermeiden.
DUMP-Veröffentlichungen möchten wir in Zukunft nur als Ausnahme gelten lassen, u. a. aus Platzgründen.
Abschließend noch einige Ausführungen zur Prüfung der der Redaktion für eine Veröffentlichung eingereichten Software. Jedes Programm wird vor seinem Abdruck in MP begutachtet bzw. getestet. Wie jeder weiß, werden einige Fehler oft erst später sichtbar. Deswegen hat der Autor in erster Linie die Verantwortung für das fehlerfreie Funktionieren eines Programmes. So versteht es sich – im Interesse aller Leser – von selbst, nur Lösungen einzureichen, die vom Autor ausgiebig getestet wurden.

✉ Computerkabinett

In unserem Betrieb soll ein Computerkabinett eingerichtet werden. Die Ausstattung soll vorerst aus einem Büro- und einem Personalcomputer bestehen. Zur Gestaltung des Raumes haben wir folgende Fragen:

- Welches Raumklima (Temperatur, Feuchte) wird verlangt?
- Sind Schallschutzmaßnahmen erforderlich? Wenn ja, welche?
- Welche elektrischen Anschlußwerte je Computer sind notwendig?
- Wie hoch ist der Platzbedarf?

usw.

Seidel
VEB Bau Grimmen, Produktionsbereich Berlin, Klement-Gottwald-Allee 292, Berlin, 1120

Leider können wir zu den von Ihnen aufgeworfenen Problemen keine detaillierte Auskunft geben. Da sie aber sicher von allgemeinem Interesse sind, veröffentlichen wir Ihre Fragen an dieser Stelle. Vielleicht erhalten Sie von anderen Lesern die gewünschten Informationen.

✉ Rätsel um Pi

Für mich war es eine ausgesprochene Freude, daß auch Sie vom Fach, wie viele Tageszeitungen, auf den π-Trick hereingefallen waren (MP 6/87, 2. Umschlagseite). Unglücklicherweise war meine Entgegnung auf die Veröffentlichung bereits im Heft 5/87 der „Technischen Gemeinschaft“, Seite 2 erschienen.
Ich hoffe, Sie verstehen den Spaß.

Prof. Dr. Karl-Heinz-Werler, Magdeburg

Wir bedauern den Fehler und bitten unsere Leser um Nachsicht. Die Anzahl der Stellen müßte richtig lauten: 133 554 000.

Termine

Erste Z1013-Tagung

WER? IG Heimcomputer der KDT am Informatikzentrum der TU Dresden
WANN? 5. Dezember 1987 von 10 bis 15 Uhr
WO? Dresden, Informatikzentrum
WAS?
– Hard- und Software zum Z1013
– Softwarebörse
WIE? Teilnahmemeldungen mit Softwareangebot und/oder Angaben zu Vortragsmeldungen (Thema, kurze Inhaltsangabe, Dauer des Referates) sind bis zum 31. Oktober 1987 an Rainer Brosig, Wilhelm-Florin-Str. 2c, Coswig, 8270, zu richten.
Prof. Dr. Tschoppe

✉ Nutzung von Zeichenkettenvariablen zum Zwischenspeichern numerischer Daten

Mit Interesse habe ich den Artikel *Nutzung von Zeichenkettenvariablen zur gleichzeitigen Abspeicherung verschiedener Werte* im Heft 5 gelesen.
Die vorgestellte Lösung hat für größere Wertbereiche der abzuspeichernden Daten eine gewisse Bedeutung, ist aber für kleinere Wertebereiche (0...255) nicht sehr effizient in der Speicherauslastung. Es ist in REDABAS/dBASE II nicht nur die Anzahl der Speichervariablen sondern auch der durch die Variablen belegbare Speicherplatz begrenzt (etwa 1 KByte). Diese Grenze ist bei der Nutzung von Speichervariablen zur Zwischenspeicherung sehr schnell erreicht.
Als Alternative zur STR- und VAL-Funktion zur Umwandlung numerischer Daten in Zeichenketten und umgekehrt bietet sich die CHR- und ORD-Funktion an (Unter dBASE RANK = ORD). Es lassen sich mit Hilfe der CHR-Funktion Werte von 0...225 in nur einem Byte einer Zeichenkette abspeichern und über die ORD-Funktion wieder in den numerischen Wert wandeln. Das Bild zeigt in Anlehnung an o. g. Artikel ein Beispielprogramm, welches genau das gleiche leistet, aber nur eine Speichervariable von 99 anstatt 198 Zeichen erfordert und die maximale Anzahl der Personen auf 255 erhöht.
Wolfgang Neumann, Berlin

```
* Initialisierung Msumm

SET TALK OFF
STORE CHR(Ø) TO Msumm
STORE 1 TO Z
DO WHILE Z<99
   STORE Msumm+CHR(Ø) TO Msumm
   STORE z+1 TO z
ENDDO

* Sequentielles zaehlen.

USE demodat
DO WHILE .NOT. EOF
   STORE CHR(ORD($(Msumm,n,1))+anzahl) TO My
   IF n=1
      STORE My+$(Msumm,2,98) TO Msumm
   ELSE
      IF n=99
         STORE $(Msumm,1,98)+My TO Msumm
      ELSE
         STORE $(Msumm,1,n-1)+My+$(Msumm,n+1,99-n) TO Msumm
      ENDIF
   ENDIF
   SKIP
ENDDO
USE

* Auslesen aus Variablen

SET PRINT ON
? '*** Altersstruktur ***'
STORE 1 TO z
DO WHILE z<1ØØ
   STORE ORD($(Msumm,z,1)) TO My
   IF My<>Ø
      ? 'Alter '+STR(z,2)+': '+STR(My,3)+' Personen
   ENDIF
   STORE z+1 TO z
ENDDO
SET PRINT OFF
```

Software-Bank der RGW-Länder

Die RGW-Länder haben mit der Einrichtung einer Software-Bank begonnen, um verschiedenartige Aufgaben mittels EDV effektiver lösen zu können. Der bisher schon rege Austausch von Software-Entwicklungen wird durch den Interfap genannten Datenfonds nun auf einheitlicher organisatorischer, rechtlicher und ökonomischer Grundlage gestellt und weiter gefördert.
Jedes Mitgliedsland kann durch den Verkauf eigener und durch den Kauf von Software anderer Zeit, Material und Arbeitskräfte sparen.
In der DDR ist der VEB Datenverarbeitungszentrum Dresden der Mittler zu den Partnern in den anderen sozialistischen Ländern. Die Einrichtung der ersten internationalen Software-Bank erfolgt im Zuge der Realisierung des Komplexprogramms des wissenschaftlich-technischen Fortschritts der RGW-Länder bis zum Jahr 2000. *ADN*

Von der Technischen Hochschule zur Technischen Universität

Kurz nacheinander erhielten die Technische Hochschule Karl-Marx-Stadt und die Technische Hochschule „Otto von Guericke" Magdeburg den Status von Technischen Universitäten. Sie waren 1952 bzw. 1953 zur Förderung von Wissenschaft und Technik gegründet worden. Ständig wurde ihr Profil erweitert, neue Wissenschaftszweige und Studienrichtungen kamen hinzu, so daß sie schon geraume Zeit außer ihrem polytechnischen weitgehend interdisziplinären Charakter tragen. Kennzeichnend dafür sind neben den traditionellen, breit gefächerten Technikwissenschaften die leistungsstarken natur- und gesellschaftswissenschaftlichen Disziplinen in Lehre und Forschung.
Dieser Entwicklung wurde mit der Verleihung des Status der Technischen Universität entsprochen. Zugleich soll damit eine neue Qualität der Wissenschaften gefördert und dokumentiert werden. Die lateinische Bezeichnung „universetas literarum" – Gesamtheit der Wissenschaften – drückt dies aus. Hohe wissenschaftliche Ergebnisse und eine über Jahre währende Profilierung der Wissenschaftsdisziplin waren hierfür wesentliche Voraussetzungen. Auch künftig bilden die Technikwissenschaften, daher Technische Universität, profilbestimmenden Schwerpunkt.

Dr. W. Kottowski

Fehlersuchsystem

Ein leistungsstarkes Meßsystem zum Feststellen und Orten von Fehlern in Anlagen oder Baugruppen der Mikroelektronik hat der VEB Elektronik Gera vorgestellt. Dieser Logikanalysator ergänzt die Mikrocomputer-Familie MC 80. Die für jedes Bauelement charakteristischen Signale oder Impulse werden auf einem Bildschirm grafisch dargestellt, so daß der Nutzer im Vergleich mit den Service-Unterlagen bei sehr geringem Zeitaufwand Abweichungen bemerken und lokalisieren kann. Dabei wirkt sich günstig die Möglichkeit aus, bis zu 32 Signale zu speichern und bei Bedarf abzurufen. Herkömmlich werden für solche Zwecke mit höherem Zeitbedarf und geringerer Treffsicherheit Oszillographen eingesetzt. Grundlage des neuen Erzeugnisses war eine Entwicklung von Wissenschaftlern der Technischen Universität Dresden, die im Betrieb modifiziert und technisch auf den aktuellen Stand gebracht wurde.
Der Logikanalysator wird in den folgenden Monaten in mehreren Wissenschaftseinrichtungen sowie in Kombinaten wie Carl Zeiss JENA und Narva Berlin getestet. In diesem Jahr sollen rund 200 Geräte gefertigt und ausgeliefert werden. Der Produktionsumfang soll im folgenden Jahr erhöht werden.

ADN

Zentraler Nachweisspeicher Angewandte Mikroelektronik

Der ZNAM speichert Informationen über Einsatzfälle der Mikroelektronik/Elektronik aus der gesamten Volkswirtschaft und stellt sie an Interessenten für Nachnutzungsentscheidungen auch wieder bereit. Er übt somit eine Vermittlerrolle zwischen Ursprungsbetrieben und Nachnutzern aus.
Rechte und Pflichten aller Betriebe und Einrichtungen sind im Gbl. I, Nr. 22 vom August 1983 verzeichnet. Darüber hinaus sind Rückfragen beim VEB Applikationszentrum Elektronik Berlin (AEB) telefonisch unter 4300811 möglich.
Neben dem vom VEB AEB herausgegebenen ZNAM-Katalog sei auf die Bezugsmöglichkeit von Arbeitsrecherchen aus dem ZNAM hingewiesen, von denen wir auszugsweise, entsprechend dem Profil der MP, die folgenden ansprechen:
AR 10/86 Zusatzbaugruppen für Mikrorechnersysteme
AR 11/87 Zusatzbaugruppen für Mikrorechnersysteme (1. Ergänzung, in Vorb.)
AR 20/86 Einsatz von Mikroprozessoren/Mikrorechnern in der Volkswirtschaft
AR 30/87 ME-Einsatz zur rationellen Energiegewinnung.

Neuer Informationsdienst

Der VEB Applikationszentrum Elektronik Berlin gibt ab 1987 als neue Informationsleistung ein Verzeichnis technischer Unterlagen – Aktive Bauelemente – heraus.
Es enthält für rund 4500 aktive elektronische Bauelemente den Nachweis, in welcher der technischen Unterlagen ZAK, Standard, Datenblattsammlung, Taschenbuch, Unterlagen auf Mikroplanfilm ausführliche Bauelemente-Dateninformationen zum jeweiligen Bauelement verfügbar sind.
Das Verzeichnis ist eine alphanumerisch geordnete Bauelementeübersicht auf der Grundlage des gültigen Sortiments mit jeweils mindestens einem Nachweis der o. g. Unterlagen. Der Preis des regelmäßig erscheinenden Informationsdienstes beträgt 25 M.
Bestellungen für gesellschaftliche Bedarfsträger (als Einzelleistung oder im Abonnement) sind formlos zu richten an:
VEB Applikationszentrum, Elektronik Berlin, Abt. DA, Mainzer Straße 25, Berlin, 1035

MP

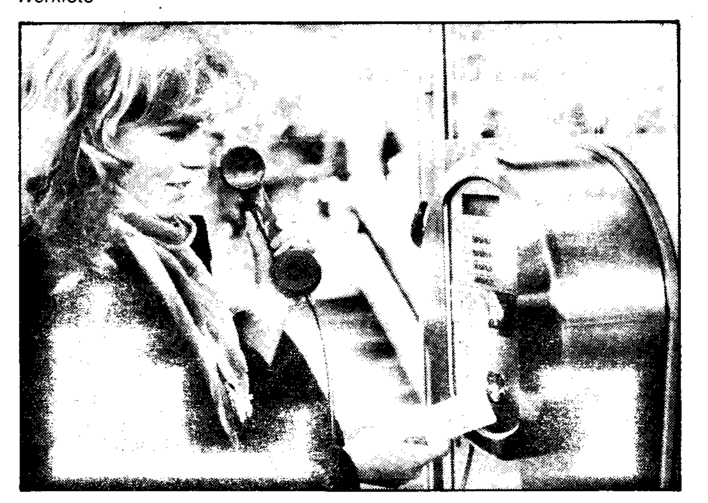
Strichcode-System

Ein einheitliches Code-System zur Identifizierung von Produkten (ETK) ist in Ungarn fertiggestellt worden. Es umfaßt 300 Gruppen, in die die wichtigsten Grundstoffe, Halbfabrikate und einige grundlegende Konsumartikel aufgenommen wurden. Zur Einführung des Systems bis Ende kommenden Jahres wurden 50 Konsultationsstützpunkte für die Betriebe errichtet. Da auch Lagerbestände mit der gleichen Computer-Codezahl versehen werden, können Erfassung, Registrierung und Veränderung erheblich beschleunigt werden. Konsumartikel werden schrittweise mit einem aus Strichkombinationen bestehenden Code versehen, der bei Exportartikeln bereits angewendet wird. Das ETK-System soll bis Ende des Jahrzehnts weiterentwickelt werden, so daß alle Materialien, Ersatzteile, Halb- und Fertigfabrikate einen einheitlichen Identifizierungscode erhalten.

„Telefon"-Karte

Bargeldloses Telefonieren an öffentlichen Fernsprechern soll das neue Kartentelefon Siemens-interset 310, das statt eines Münzeinwurfs einen Schlitz für spezielle Telefonkarten hat, ermöglichen (siehe Bild). Dazu wurde in der BRD im Dezember vergangenen Jahres ein entsprechender Großversuch gestartet. Die Elektronik im Fernsprecher bucht entweder die Telefongebühren direkt von einer Telefonkarte ab oder belastet über eine Buchungskarte das Fernmeldekonto des Anrufers, bzw. der Karteninhaber erhält eine Rechnung. Das „Gehirn" der Chipkarte soll ein winziger Speicherchip mit einer ausgeklügelten Sicherheitslogik sein.

Werkfoto



Analogwerterfassung mit den A/D-Wandlern C 571 C/C 570 C

Klaus Christen
VEB Halbleiterwerk Frankfurt (Oder)

Kurzbeschreibung der A/D-Wandler C 571 C (C 570 C)

Der C 571 ist ein kompletter, monolithisch integrierter Analog-Digital-Wandler mit einer Auflösung von 10 Bit, bestehend aus D/A-Wandler, interner Referenzspannungsquelle, Taktgenerator, Komparator, Sukzessiv-Approximations-Register und Ausgangstreibern. Durch Lasertrimmen werden die Linearität des Wandlers, die Endwertspannung und deren Temperaturkoeffizient abgeglichen. Der C 570 ist eine funktionsgleiche Version des C 571, für die nur eine Auflösung von 8 Bit garantiert wird.
Das Blockschaltbild des A/D-Wandlers C 571 ist in Bild 1 dargestellt. Die Bauelemente benötigen zwei Betriebsspannungen von +5 V und −15 V. Die A/D-Wandler verarbeiten im Unipolarbetrieb Eingangsspannungen zwischen 0 V und +10 V und im Bipolarbetrieb zwischen −5 V und +5 V.
Die Wahl der Betriebsart erfolgt durch Verbinden von Pin 15 mit Analogmasse (Unipolarbetrieb) bzw. durch Nichtbeschalten von Pin 15 (Bipolarbetrieb). Die Bauelemente besitzen einen niedrigen Eingangswiderstand mit einem typischen Wert von 5 kOhm. Der L/$\overline{S}$-Eingang (Löschen/$\overline{\text{Starten}}$ und die Ausgänge des Wandlers sind TTL-kompatibel.

Zeitverhalten

Der zeitliche Verlauf des Umsetzungsvorganges ist in Bild 2 dargestellt. Im gelöschten Zustand (L/$\overline{S}$ = H) sind die Tristate-Datenausgänge hochohmig, und der Statusausgang STS liegt auf H-Potential. Mit dem Übergang von H- auf L-Potential am L/$\overline{S}$-Eingang wird der Umsetzzyklus ausgelöst. Die Beendigung des Umsetzzyklus wird durch den Übergang auf L-Potential am STS-Ausgang signalisiert. Nach einer Verzögerung von 0,5 µs sind die Daten an den aktivierten Datenausgängen gültig. Die Umsetzzeit liegt zwischen 15 und 40 µs, typisch bei 20 µs. Maximal 1,5 µs nach dem L-H-Übergang am L/$\overline{S}$-Eingang nimmt der Statusausgang H-Potential an, und die Datenausgänge gehen in den hochohmigen Zustand über. Wird der L/$\overline{S}$-Eingang wieder auf L-Potential geschaltet, beginnt eine erneute Umsetzung. Die minimale Impulsbreite für das Abschalten der Datenausgänge und für den erneuten Umsetzstart beträgt 2 µs.
Ein positiver Impuls am L/$\overline{S}$-Eingang mit tp ≧ 2 µs während des Umsetzens bewirkt den Abbruch der laufenden Umsetzung und einen erneuten Umsetzstart.

Mikrorechneranpassung

Die A/D-Wandler C 571 (C 570) können mit einem Minimum an zusätzlichen Bauelementen von einem Standard-Mikroprozessor gesteuert bzw. an deren Datenbus angepaßt werden. Einen Schaltungsvorschlag für die Zusammenschaltung des C 571 (C 570) mit dem Mikroprozessor U 880 zeigt Bild 3. Der Umsetzvorgang wird gestartet, wenn die Steuerleitungen $\overline{IORQ}$, $\overline{WR}$ = L und der Ausgang des Adreßdekoders ADR 1 = H sind (E/A-Schreiben). Ein monostabiler Multivibrator sichert dabei die notwendige Startimpulsbreite. Wenn die Steuerleitungen $\overline{IORQ}$, $\overline{RD}$ = L und der Ausgang des Adreßdekoders ADR 1 = H sind (E/A-Lesen), werden BIT 1 ... 8 über die Bustreiber (DL 541) auf den Datenbus gegeben. Kommt ein 10-Bit-A/D-Wandler (C 571) zur Anwendung, werden BIT 9, 10 über einen weiteren Bustreiber bei $\overline{IORQ}$, $\overline{RD}$ = L und ADR 2 = H an den Datenbus geschaltet. Zusätzlich kann über diesen Bustreiber der Statusausgang des Wandlers abgefragt werden (Polling), um das Ende des Umsetzvorganges festzustellen.
Eine zusätzliche Zwischenspeicherung der Ausgangsdaten ist nicht notwendig, da bis zu einem erneuten Umsetzstartkommando die Daten vom C 571/C 570 gehalten werden.

Eine weitere Variante zur Anschaltung eines A/D-Wandlers an den Datenbus eines Mikroprozessors besteht in der Verwendung von Peripherie-Interface-Schaltkreisen, z. B. der PIO U 855. Bild 4 zeigt einen Schaltungsvorschlag zur Anschaltung von maximal 8 C 570 im Multiplexbetrieb.
Die Tristate-Datenausgänge der A/D-Wandler erlauben die ausgangsseitige Parallelschaltung mehrerer Wandler. Das 8-Bit-Datenwort eines Wandlers kann über Kanal A der PIO eingelesen werden (Byteeingabe). Über Kanal B der PIO und die L/$\overline{S}$-Eingänge werden die Wandler gesteuert (Byteausgabe).
Der Umsetzvorgang nur eines Wandlers wird durch den Übergang auf L-Pegel an dessen L/$\overline{S}$-Eingang gestartet. Nach Beendigung des Umsetzvorganges werden nur die Datenausgänge dieses Wandlers aktiv.
Die H/L-Flanke des Statussignals eines A/D-Wandlers bewirkt die Erzeugung eines Strobe-Impulses (ca. 1 µs), mit dem das Datenbyte in das Eingaberegister des PIO-Kanals A geladen wird und dessen L/H-Flanke eine Interruptanmeldung (falls die Voraussetzungen dafür gegeben sind) auslöst.
Mit Hilfe eines Interruptbedienprogramms kann der nächste Wandler gestartet und das Datenbyte in die CPU übernommen werden.

10-Bit-Datenerfassungssystem

Datenerfassungssysteme werden benötigt, um eine Vielzahl von analogen Signalen in digitale Werte zur Weiterverarbeitung in einem Rechner aufzubereiten. Aus der möglichen Variantenvielfalt wird ein aus mehreren Einzelkomponenten aufgebautes 16-Kanal-10-Bit-Datenerfassungssystem vorgestellt (Bild 5):

- Multiplexer:

Mit dem BiFET-Analogmultiplexer **MAB 16 E** (Tesla) wird einer der 16 Analogeingänge in Abhängigkeit von der angelegten binären 4-Bit-Adresse an den nachfolgenden Abtast-/Halteverstärker geschaltet. Das Umschalten der Eingänge beim Anlegen einer Kanaladresse geschieht konfliktlos, da die Ausschaltzeit der Analogschalter geringer als deren Einschaltzeit ist. Der „Ein"-Widerstand von weniger als 400 Ohm ist für die hier geforderten Eingangsspannungsbereiche von −5 V ... +5 V und 0 ... +10 V konstant. Die positive Eingangsspannung sollte auf ≦ 11 V begrenzt werden. Die vom Daten- oder Adreßbus des Mikrorechners bezogene Kanaladreßinformation wird im Adreßregister (DL 175) zwischengespeichert.

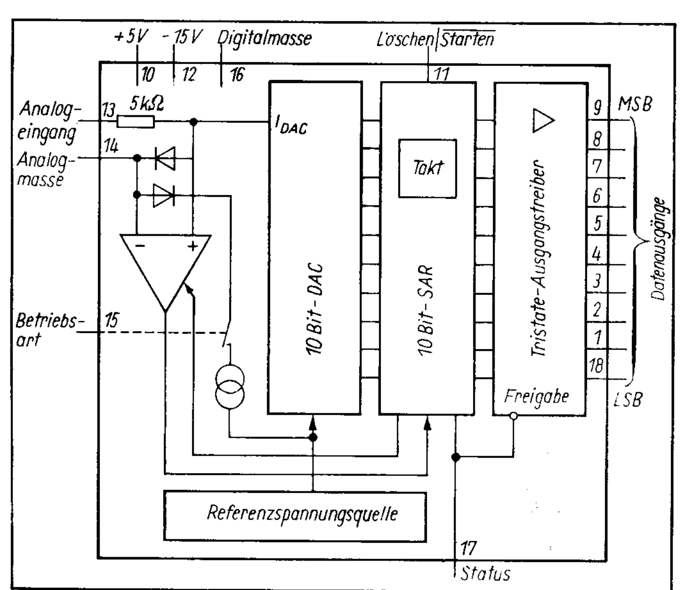


Bild 1 Blockschaltbild des A/D-Wandlers C 571

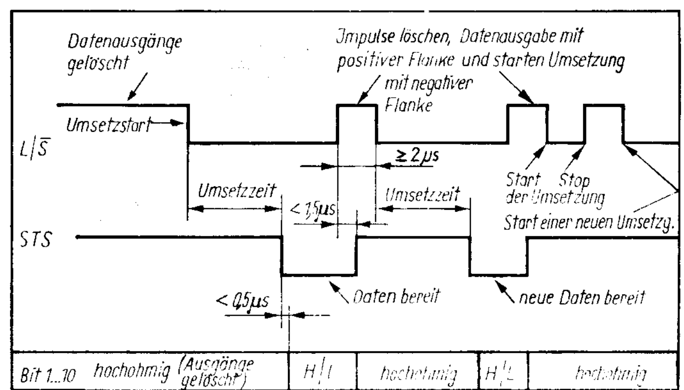


Bild 2 Zeitlicher Verlauf des Umsetzvorganges

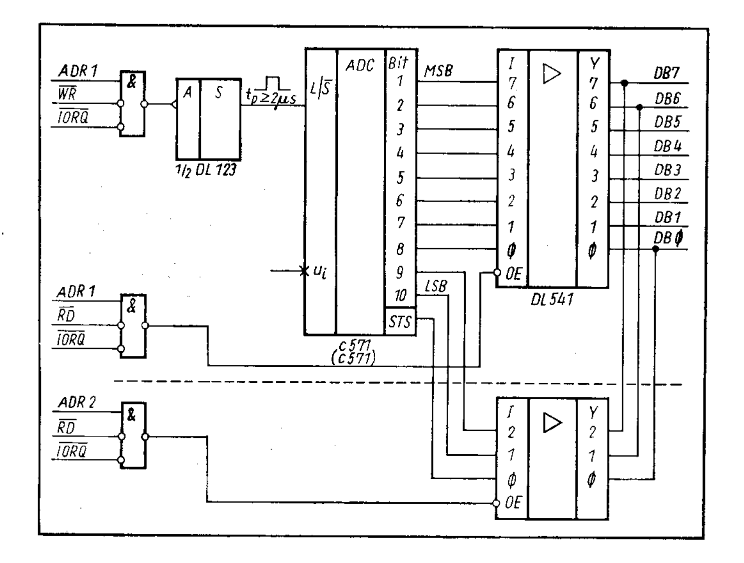


Bild 3 Zusammenschaltung des C571 mit Mikrorechner auf U880-Basis

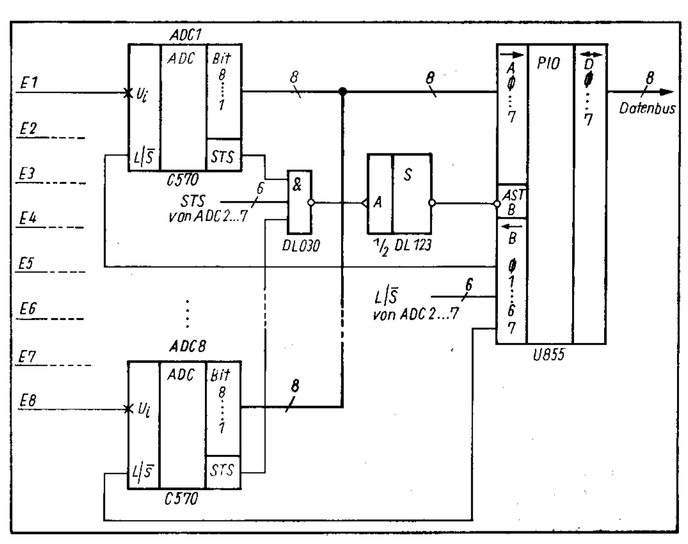


Bild 4 Mikrorechneranpassung von acht C571 (Multiplexbetrieb) mittels PIO U855

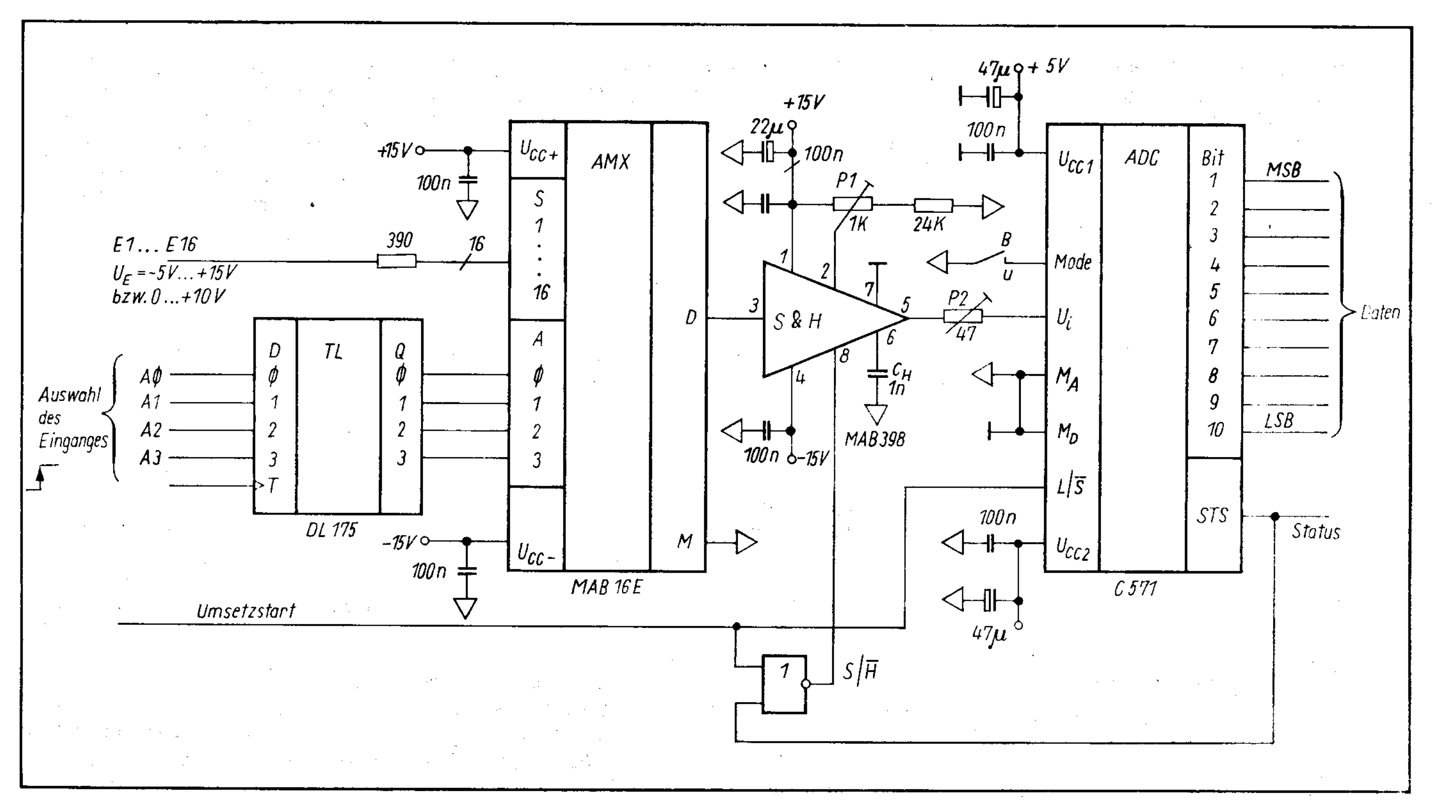


Bild 5 16-Kanal-10-Bit-Datenerfassungssystem

• Abtast-/Halteverstärker:

Die Digitalisierung sich schnell ändernder Analogsignale erfordert einen Abtast-/Halteverstärker (AHV). Er speichert das Eingangssignal für die Dauer der Umsetzung und dient gleichzeitig als Impedanzwandler. Die Abtast-/Halteschaltung MAB 398 (Tesla) erfaßt einen 10-V-Eingangssignalsprung mit der benötigten 10-Bit-Genauigkeit in 4 μs. Die Wandlungszeit pro Kanal beträgt damit 34 μs (Erfassungszeit tacq = 4 μs, Umsetzzeit tc, max. = 30 μs).
Als Haltekondensator wird ein 1-nF-Polystyrol-Kondensator verwendet.

Wird eine Umsetzung am A/D-Wandler gestartet, geht der AHV in der im Bild 5 gezeigten Konfiguration automatisch für die Dauer der Umsetzung in den „Halte"-Zustand über. Mit dem Spindeleinstellregler P 1 wird der Nullpunkt für das gesamte System eingestellt.

• A/D-Wandler:

Mit der Wahl der Betriebsart des Wandlers wird der Eingangsspannungsbereich des Datenerfassungssystems von −5 V ... +5 V oder von 0 ... +10 V festgelegt.
Der Endwertabgleich für das gesamte System wird mit dem Spindeleinstellregler P 2 vorgenommen. Die +5-V-Betriebsspannung ist gegen Digitalmasse und die -15-V-Betriebsspannung ist gegen Analogmasse jeweils nahe am Schaltkreis mit 47 μF und 100 nF (keramischer Scheibenkondensator) abzublocken. Um Verkopplungen zwischen den analogen und den störbehafteten digitalen Signalen zu vermeiden, sind separate Masseführungen zu verwenden. Alle Masseleitungen sind sternförmig – mit dem Analogmasseanschluß des A/D-Wandlers als Mittelpunkt – zu verbinden. Zu diesem Punkt wird auch die Digitalmasse geführt.

Systemumsetzzyklus

Die Wandlung der Eingangsspannung an den Analogkanälen geschieht im Normalfall in folgender zeitlicher Reihenfolge:
1. Nach der Ausgabe/Übernahme der Kanaladresse des anzuschaltenden Eingangs muß weiterhin die Zeit für das Einschwingen des Signals am Eingang des AHV berücksichtigt werden.
2. Mit der Ausgabe eines positiven Impulses von tp ≧ 2 μs an den Eingang „Umsetzstart" wird der Umsetzvorgang gestartet.

Fortsetzung auf Seite 262

Zeitoptimierte A/D-D/A-Baugruppe mit C 571/C 565

Thomas Schmidt, Torsten Greiner
Technische Hochschule Leipzig,
Sektion Automatisierungsanlagen

Einführung

Mit dieser Baugruppe sollen bei einer Auflösung von 8 Bit schnell veränderliche analoge Signale erfaßt und analogdigital umgesetzt werden, so daß sie auf einer Rechnerschnittstelle (z. B. EMR) verfügbar sind. Gleichzeitig, d. h. im time-sharing, soll die Ausgabe und D/A-Wandlung von im Rechner verarbeiteten Werten über die gleiche Schnittstelle gewährleistet werden. Die Anwendungsgebiete erstrecken sich von der μR-gesteuerten Signalverarbeitung (z. B. NF-Signale) über die intelligente Sensorik (digitale Kennlinienkorrektur/Störgrößenkompensation) bis zur Prozeßsteuerung. Damit ist eine schnelle Reaktion auf die sich ändernde Prozeßgröße möglich, so daß z. B. die Geschwindigkeit einer durch Mikroprozessor gesteuerten Regelung beachtlich gesteigert werden kann.
Die Schaltung stellt eine Applikation mit der nach dem Verfahren der sukzessiven Approximation arbeitenden IS C 571 D dar /1/, /2/. Da für zeitoptimale Verarbeitung bei 8-Bit-μR-Systemen die Umsetzungsbreite auf 8 Bit beschränkt bleibt, ist in diesem Falle auch der Einsatz des Anfalltyps C 570 D möglich. Dabei ist jedoch zu beachten, daß auch die Umsetzzeit beim C 570 größer sein kann /2/, so daß es in jedem Fall von Vorteil ist, ausgemessene Exemplare für zeitkritische Anwendungen einzusetzen. Für den D/A-Wandler genügt der Anfalltyp C 5658 D.

Thomas Schmidt (24) studiert nach seiner Berufsausbildung mit Abitur als E-Monteur seit 1982 an der Technischen Hochschule Leipzig, Fach Automatisierungstechnik.
Er ist Beststudent, bereits seit dem 1. Studienjahr Hilfsassistent und Mitglied in zwei Jugendforscherkollektiven.
Dipl.-Ing. Torsten Greiner (29) studierte an der Technischen Hochschule Karl-Marx-Stadt (jetzt Technische Universität Karl-Marx-Stadt), Sektion Informationstechnik, im Fach Gerätetechnik. Seit 1983 ist er als wissenschaftlicher Assistent an der Technischen Hochschule Leipzig tätig und arbeitet gegenwärtig an einer Dissertation auf dem Gebiet der intelligenten Sensorik.

Die Umsetzzeit des C 571 von 15–30 μs (exemplarabhängig) wird als guter Kompromiß zwischen aufwendigen schnellen Parallel-A/D-Wandlern und hochgenauen, langsameren integrierenden A/D-Wandlern angesehen, da eine weitere Verringerung der Umsetzzeit die notwendige Zeit zur Verarbeitung im Mikrorechner bei Ausnutzung der maximal möglichen Umsetzrate ebenfalls verkürzen würde. Diese zur Eingabe, Ausgabe, Modifikation der Steuersignale und Datenverarbeitung benötigte Zeit liegt aber bereits an der Grenze der durch die Baugruppe ermöglichten Umsetzzeit von 20–35 μs, so daß auf höhere Abtastgeschwindigkeiten zugunsten eines minimalen Hardwareaufwands verzichtet wurde.
Für sich schnell ändernde analoge Signale (du/dt ≧ 0,4 V/ms) macht sich eine Sample-and-Hold-Schaltung erforderlich /3/, um die Spannung am Eingang des A/D-Wandlers während der Umsetzzeit konstant zu halten, da ansonsten ein unvertretbar hoher Meßfehler auftreten würde (zulässige Abweichung der Analogspannung am C 571 während der Umsetzzeit beträgt 1/4 LSB ≙ 10 mV).

Funktionsbeschreibung

Zunächst soll anhand Bild 1 die prinzipielle Arbeitsweise der Baugruppe erläutert werden. Um sie so universell wie möglich einsetzen zu können, wurde ein empfindlicher Eingangsspannungsbereich von 0 ... 0,1 V vorgesehen. Auf Grund des hohen Eingangsspannungsbedarfes des C 571 von 10 V macht sich eine Vorverstärkung des Signals um $V_u = 100$ notwendig, die sich in 2 Stufen zu je $V_u = 10$ aufteilt. Zwischen diesen Verstärkerstufen befindet sich die Sample-and-Hold-Schaltung, da mit einer solchen Anordnung ein gutes Einschwingverhalten erreicht wird. Die erste Stufe (N1) bildet einen hochohmigen Eingang ($\approx 10^{12}\,\Omega$) und ermöglicht gleichzeitig das niederohmige Laden des Speicherkondensators. Dieser gewährleistet zusammen mit dem Tortransistor (SFET) ein Speichern der Analogspannung für die Zeit der A/D-Umsetzung (Sample-and-Hold-Schaltung). Die zweite Verstärkerstufe (N2) sichert die notwendige hochohmige Abtastung des Speicherkondensators und stellt die so aufbereitete Eingangsspannung dem A/D-Wandler zur Verfügung.
Durch eine LH-Flanke am Steuertakt c, der vom Rechner programmtechnisch realisiert werden kann, wird mit dem Strobe STB 1 der in dem zurückliegenden Meßzyklus gewonnene Wert in das Latch RG 1 eingeschrieben, der notwendige Lösch-Startimpuls ($L/\overline{S}$) von 2 μs für den C 571 ausgelöst, die Sample-Phase abgebrochen und in den Hold-Zustand übergegangen. Nach Beendigung der Umsetzung durch den ADU geht das Statussignal $\overline{STS}$ nach L und die Sample-and-Hold-Schaltung öffnet sofort wieder, um den Haltekondensator auf den aktuellen Spannungswert von N1 nachzuladen.
Während der Umsetzzeit kann das Steuersignal Output Enable ($\overline{OE}$) zum Einlesen des Wertes von RG 1 in den Rechner Low gesetzt werden. Nachdem der Wert eingelesen wurde, kann der Bus mit einem vom Rechner gesendeten Wert beschrieben werden ($\overline{OE}$ wieder H!), der anschließend mit HL-Flanke des Taktes c durch den Strobe STB 2 in das Latch RG 2 eingeschrieben wird. Gleichzeitig wird durch den D/A-Wandler dieser 8-Bit-Wert umgesetzt und durch N3 im Bereich von 0 ... 10 V am Analogausgang bereitgestellt.
Den notwendigen Überblick über den zeitlichen Ablauf aller Steuersignale und Daten im time-sharing gibt Bild 2. Die Organisation des Datenverkehrs ist bei richtiger Nutzung der Steuersignale $\overline{OE}$ und c in weiten Grenzen variierbar, jedoch muß beachtet werden, daß $\overline{OE}$ ständig H sein muß, wenn Daten vom Rechner gesendet werden (PIO als Ausgabeport initialisiert).

Analoger Eingangsteil/Sample-and-Hold-Stufe

Diese Stufe ist für den Zeitverlauf die kritischste, da sie die Eingangsspannung nicht verfälschen darf (Einschwing- und Koppelprobleme) und außerdem eine $V_u = 100$ mit einem Fehler von ≦ 0,2% realisieren muß (für 8 Bit ausreichende Genauigkeit). Sie ist in Bild 3 dargestellt.
Als OPV kommen Typen mit SFET-Eingangsstufen zur Anwendung. N1 arbeitet als 1. Verstärkerstufe und gleichzeitig als Impedanzwandler, VD_1 und VD_2 sichern den notwendigen Eingangsspannungsschutz bis U_I ≦ 1000 V. Als Tortransistor der Sample-and-Hold-Stufe wird ein SFET KP 303A mit $U_p < 2$ V verwendet. Die Höhe der Gatespannung in der Hold-Phase beträgt etwa $U_{GS} \approx -4$ V und wird durch die Transistorstufe VT_4 und VT_5 geschaltet. In der Sample-Phase wird die Gate-Spannung von VT_1 über den Widerstand von 3,3 K der Sourcespannung mitgeführt, so daß bei $U_{GS} = 0$ optimale Leitfähigkeit erzielt wird und somit ein niedriger Lade- und Entladewiderstand für den Haltekondensator C_H gesichert ist. Mit dem komplementären Impuls am Source des VT_1 wird über R2 und VT_3 die durch die Gate-Kanal-Kapazität und den Schaltimpuls über VT_4 in den Hold-Kondensator eingespeiste Ladung kompensiert. Der Hold-Kondensator von 220 pF sichert bei der maximal nötigen Haltezeit von 30 μs (Umsetzzeit des C 571) eine decary rate von 0,001 % der zu messen-

---

(Fortsetzung von Seite 261)
3. Der AHV wird in den „Halte"-Zustand geschaltet, und vom Statussignal wird mit H-Pegel die laufende Umsetzung angezeigt.
4. Da für die Dauer des Umsetzzyklus die Analogeingangsspannung vom AHV gehalten wird, kann inzwischen die Kanaladresse des nächsten anzuschaltenden Eingangs vom Adreßregister übernommen werden. Das Umschalten des Analogmultiplexers bleibt also ohne Einfluß auf die Umsetzdauer pro Kanal.
5. Die Beendigung des Umsetzzyklus wird durch den Übergang auf L-Potential am Statusausgang signalisiert. Der AHV geht in den „Abtast"-Zustand über.
. Nach einer Erfassungszeit von ≧ 4 μs kann das nächste Umsetzstartkommando ausgegeben werden. Zwischenzeitlich können die A/D-Wandler-Daten vom Mikroprozessor gelesen werden.

Literatur

/1/ Fachbereichsstandard Entwurf 9/85. 10-Bit-Analog-Digital-Wandlerschaltkreise C 571 D, C 571 D1 und 8-Bit-Analog-Digital-Wandlerschaltkreis C 570 D
/2/ Informationsblatt C 570 D, C 571 D des VEB HFO

den Spannung und eine Sperrdämpfung bei einer Koppelkapazität des VT_1 von 0,05 pF von 36 dB. Die durch das Einschwingverhalten begrenzte Sample-Zeit der Schaltung beträgt ca. 3,5 µs. Der OPV N2 sichert mit einem Eingangswiderstand von $\approx 10^{12}\Omega$ die hochohmige Ankopplung an den Speicherkondensator und realisiert die zweite Verstärkerstufe. Die Gesamtverstärkung von N1 und N2 ist durch R_4 zu $V_u = 100$ abgleichbar.

Digitalteil/Steuerung

Die Steuerung des A/D-D/A-Wandlers erfolgt über die Taktleitung c (Bild 4). Bei der LH-Flanke wird das Strobe-Signal STB 1 als Nadelimpuls von etwa 60 ns generiert (D 2.4, D 2.3, D 1.3 u. D 2.2), bei dessen Vorderflanke das Latch 1 die bei der letzten Messung gewonnenen Daten übernimmt und je nach Bedarf (Signal $\overline{OE}$) an der Schnittstelle zur Verfügung stellt. Durch die Rückflanke des STB 1 wird das Monoflop (D 3) gestartet und erzeugt das Lösch-/Startsignal $L/\overline{S}$ von 2 µs Länge. Damit beginnt eine neue Messung, und es erfolgt über D 1.2, D 1.1 bei $L/\overline{S}$ = L das Umschalten in die Hold-Phase ($S/\overline{H}$ = L). Sobald die Ausgänge des C 571 gelöscht sind, wird $\overline{STS}$ = H und damit über D 2.1. die Hold-Phase während der Umsetzung weiter beibehalten.
Ist die Umsetzzeit beendet ($\overline{STS}$ = L), wird $S/\overline{H}$ = H, dadurch der Tortransistor der Sample-and-Hold-Stufe geöffnet und der Speicherkondensator nachgeladen. Dieser Zustand wird solange beibehalten, bis eine neue LH-Flanke an c eintrifft.
Bei der HL-Rückflanke an c wird der Strobeimpuls STB 2 mit D 2.5, D 1.4 und D 2.6 generiert, durch dessen Vorderflanke das auf der Schnittstelle stehende Byte in das Latch 2 eingeschrieben wird und zur D/A-Wandlung durch den C 5658 bis zum Eintreffen der nächsten HL-Flanke zur Verfügung steht. N3 arbeitet dabei als Strom-Spannungswandler und stellt eine Ausgangsspannung im Bereich von 0 ... 10 V bereit /4/.

Erfahrungen

Mit dem beschriebenen Modul konnte eine leistungsfähige periphere Baugruppe geschaffen werden, die sich in jedes 8-Bit-Mikrorechnerkonzept implementieren läßt. Die Anwendung im Zusammenhang mit Einchip-Mikrorechnern ist möglich und wurde vom Autor erfolgreich erprobt. Die Nutzung des Moduls in automatisierten Meßwerterfassungssystemen läßt sich mittels Ergänzen eines Eingangsspannungsteilers, der über einen Dekoder vom Rechner geschaltet wird, erweitern, so daß sich ein sehr großer Meßbereich realisieren läßt. Außerdem ist über einen Analogmultiplexer die Abtastung mehrerer Meßstellen in relativ kurzer Zeit möglich, so daß die analoge Ausgabe in Abhängigkeit mehrerer zu überwachender Größen erfolgen kann. Durch Ergänzung des Schalters S1 (Bild 3) wurde die Möglichkeit des C 571 ausgenutzt, auch bipolare Signale verarbeiten zu können ($U_I = -0{,}05\,V \ldots 0{,}05\,V$). Eine nähere Beschreibung erfolgt in /5/. In Umgebung mit starken Störeinflüssen macht sich das Abschirmen der sehr hochohmigen Eingangs- und Sample-and-Hold-Schaltung notwendig, da sonst ein Einkoppeln der Störspannung auftreten kann. Die maximal mögliche Umsetzrate ergibt sich aus der exemplarabhängigen Umsetzzeit des C 571 und den notwendigen 3,5 µs für die Sample-Phase. Mit dem Muster konnte eine Abtastung mit 40 kHz sicher realisiert werden.

Literatur

/1/ Dietrich, J.; Kahl, B: 10- und 8-bit A/D-Wandlerschaltkreise C 571 und C 570. Referate des 11. Mikroelektronik-Bauelemente-Symposiums 1985 in Frankfurt (O.) Band 2, S. 341
/2/ Datenblatt: C 570 D, C 571 D; VEB Halbleiterwerk Frankfurt (O.)
/3/ Richter, K.-H.: 8 bit A/D-Wandler für Videosignalverarbeitung. Radio, Ferns., Elektron. Berlin 34 (1985) 7, S. 418–420
/4/ Datenblatt: C 565 D. VEB Halbleiterwerk Frankfurt (O.)
/5/ Heß, Chr.: C 571 D an U 880-Systemen Mikroprozessortechnik, Berlin 1 (1987) 9, S.264

KONTAKT

Technische Hochschule Leipzig,
Sektion Automatisierungsanlagen,
Karl-Liebknecht-Straße 132, Leipzig. 7030;
Tel. 3943196

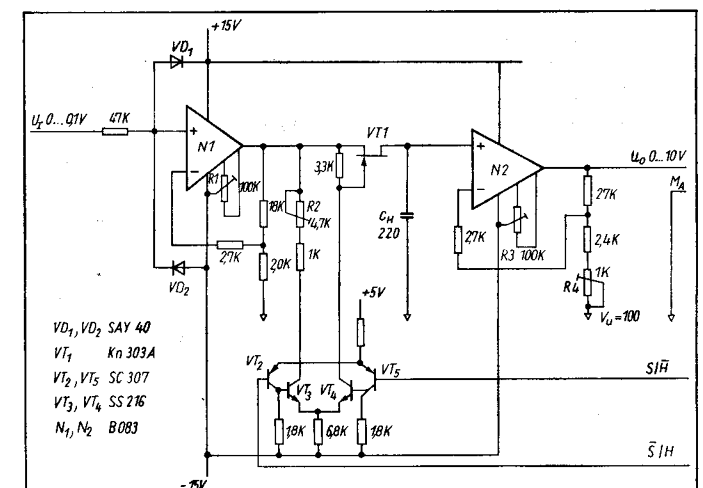
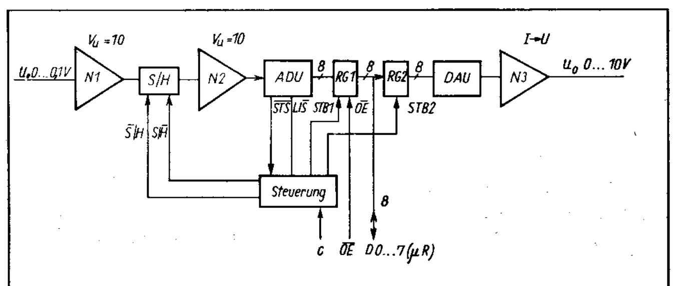
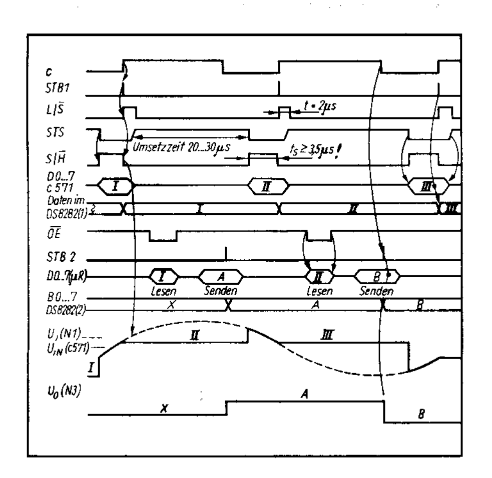


Bild 3 ▶ Stromlaufplan Analog-Eingangsteil

Bild 1 Prinzipschaltbild der Baugruppe





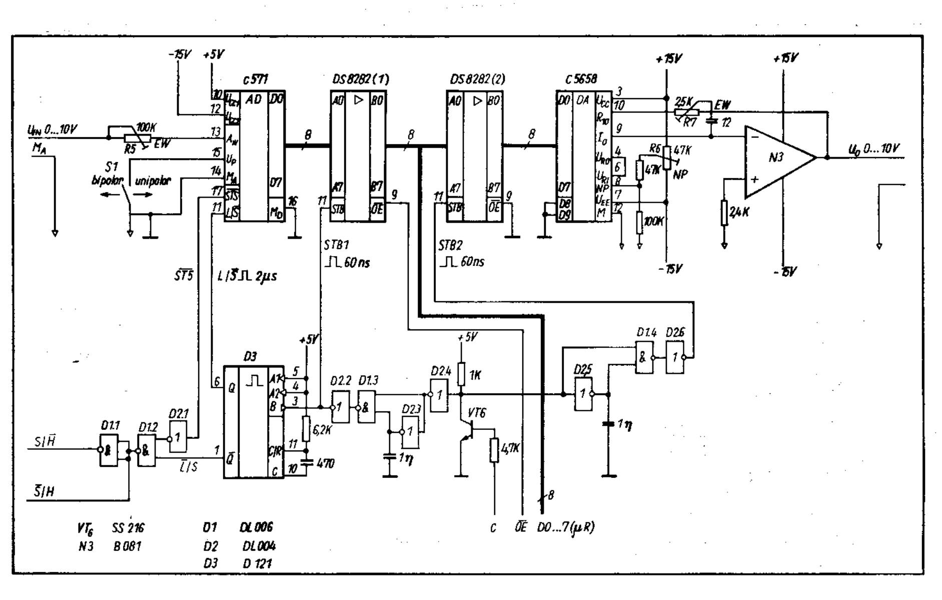


Bild 2 Zeitlicher Verlauf digitaler und analoger Signale im time-sharing
Bild 4 Stromlaufplan A/D-Wandler, Digitalteil und D/A-Wandler

C 571 D an U 880-Systemen

Christian Heß
Technische Hochschule Leipzig,
Sektion Automatisierungsanlagen

Vorbemerkung

Die digitale Meßtechnik stellt immer höhere Anforderungen an Genauigkeit und Geschwindigkeit bei der Meßwerterfassung. Der monolithisch integrierte Analog-Digital-Wandler C 571 D kommt dem mit einer Auflösung von 10 Bit und einer Umsetzzeit von 15... 30 μs entgegen. In Bild 1 ist die Anschlußbelegung des C 571 D (Vergleichstyp: AD 571 J von Analog Devices) dargestellt.

PIO im Bitbetrieb

Eine einfache Variante zur Ankopplung des C 571 D an die PIO eines Mikrorechners stellt Bild 2 dar. Für den 8-Bit-Anfalltyp des C 571 D, den C 570 D, wurde eine ähnliche Schaltung in /1/ vorgestellt. Zusätzlich ist eine Möglichkeit zum Abgleich des 10-Bit-A/D-Wandlers auf 10 mV pro LSB und somit einen Meßbereich von 10,23 V nach /2/ dargestellt.
In der gezeichneten Stellung des Schalters SW liegt die bipolare Betriebsart des Wandlers vor. Durch Schließen von SW wird zur unipolaren Betriebsart übergegangen. Die Ein- und Ausgangsinformationen für beide Betriebsarten des C 571 D sind in Tafel 1 zusammengestellt.

Abgleich

Der Abgleich kann in einer der beiden Betriebsarten erfolgen. In der unipolaren Betriebsart wird bei einer Eingangsspannung von 0 V der Dickschichteinstellregler NC verstellt, bis alle Ausgangsbits L-Pegel einnehmen. Der Dickschichteinstellregler EW wird bei U_I = 10,23 V verstellt, bis alle Bits H-Pegel eingenommen haben. Dieser Abgleichvorgang ist ggf. zu wiederholen.
Durch intern lasergetrimmte Dünnfilmwiderstände /2/ wird gewährleistet, daß der Abgleich für beide Betriebsarten gemeinsam vorgenommen werden kann.

Tafel 1 Betriebsarten des C 571 D

Betriebsart	bipolar	unipolar
Schalter SW	offen	geschlossen
Eingangsspannung U_I	−5,12 V 0 V +5,11 V	0 V +10,23 V
Daten BIT...BIT	000H 200H 3FFH	000H 3FFH

Meßwerterfassung

Durch softwaremäßiges Setzen der PIO-Leitung B3 wird der ADU in die Löschphase versetzt. Mit dem folgenden Rücksetzen von B3 wird die Wandlung gestartet. Der Statusausgang des ADU wird durch B2 abgefragt, bis L-Pegel anliegt. Ist dies der Fall, können die digitalen Daten in die CPU eingelesen werden. Bei einer Taktfrequenz des Mikrorechners von 2,5 MHz wird eine Meßzeit t_M nach Gl. (1) benötigt, bis ein digitalisierter Analogwert einem Doppelregister der CPU zur Verfügung steht.

$$t_M \geqq t_{START} + t_C + t_{SA} + t_{LESEN} = 52 \ldots 79\,\mu s \qquad (1)$$

t_M Meßzeit
t_{START} Zeit für den softwaremäßigen Start der Wandlung (= 36 Takte)
t_C Umsetzzeit des C 571 D (15 ... 30 μs)
t_{SA} Zeit für die softwaremäßige Statusauswertung (26 ... 57 Takte)
t_{LESEN} Zeit zum PIO-Lesen (30 Takte)

Ein Programm zu dieser Variante, allerdings mit dem 8-Bit-Anfalltyp C 570 D, wurde bereits in /1/ vorgestellt.

PIO im Byte-Eingabe-Mode und Auslösung der A/D-Wandlung durch Verknüpfung von CPU-Halt mit externem Synchronisationssignal SYN

Bei der Konzipierung einer schnelleren Variante der Meßwerterfassung mit dem C 571 D wurde von dem Grundgedanken ausgegangen, die Verarbeitung des Meßwertes (n−1) während der Umsetzzeit für den Meßwert (n) vorzunehmen.
Zur Realisierung wurden gemäß Bild 3 folgende Hard- und Softwarekomponenten eingesetzt:

– PIO-Initialisierung auf Byte-Eingabe;
– ein CTC-Kanal interruptfähig als Zähler bis 1;
– Verknüpfung von /HALT mit SYN löst A/D-Wandlung und am CTC eine Interruptmeldung aus;
– in Interruptserviceroutine (ISR) erfolgen die Datenübernahme aus den PIO-Eingaberegistern und die Meßwertverarbeitung;
– Statusmeldung /SA legt Strobe /STB beider PIO-Ports kurz auf L;
– Abgleichelemente des ADU wie in Bild 1;
– ggf. Sample & Hold-Schaltung vor ADU (z. B. KR 1100 SK 2, AD 583, /5/).

Bild 4 enthält die zugehörige Darstellung des Zeitverhaltens. Wenn die CPU im Halt-Zustand ist und das externe Synchronisationssignal SYN (das z. B. von einem Chopper gesendet wird) anliegt, wird das aus zwei NAND-Gattern bestehende RS-Flipflop gesetzt, und der A/D-Wandler löscht seine Daten.
Nach der Zeit $t_d \leqq 1,5\,\mu s$ schaltet das Statussignal /SA auf H, wodurch signalisiert wird, daß keine gültigen Daten anliegen, da die Datenausgänge hochohmig geschaltet sind. Durch die Gatterlaufzeiten der verwendeten CMOS-Schaltkreise, die durch kapazitive Belastung von D 1.2 mit C1 $\leqq$ 4,7 nF verlängert werden können, wird das Flipflop verzögert rückgesetzt und der A/D-Wandler gestartet. Die Verzögerung muß so groß sein, daß der Löschimpuls die Herstellerforderung /3/ nach einer Dauer $t_p \geqq 2\,\mu s$ erfüllt.
Bei Verwendung von TTL-Schaltkreisen müßte ggf. vor dem Rücksetzeingang des RS-Flipflops eine zusätzliche Flankenverzögerungsschaltung zwischengeschaltet werden.
Nach der Umsetzzeit t_c schaltet /SA wieder auf L, und /STB wird durch die Impulsverkürzungsschaltung für eine Zeit $t_{w\,(ST)} > 150$ ns auf L gelegt. Diese Zeit wird benötigt, um die Daten in die PIO-Eingaberegister einzuschreiben /4/. Mit der L/H-Flanke von /STB werden die Daten dann fixiert und stehen zur

Bild 1 Anschlußbelegung des C 571 D

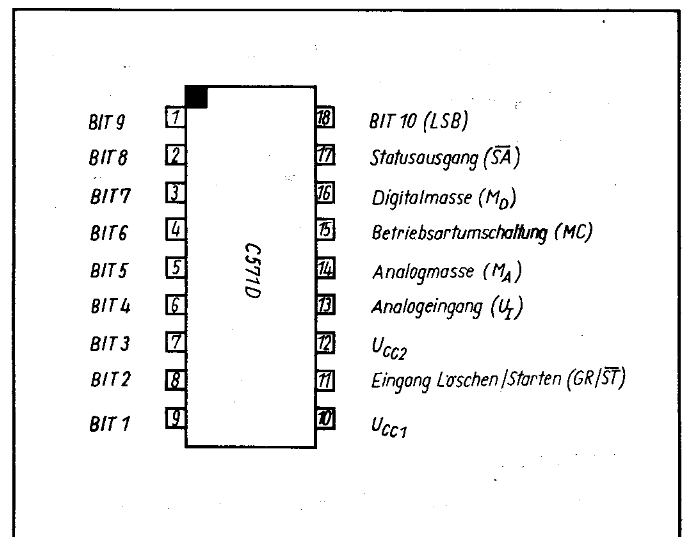
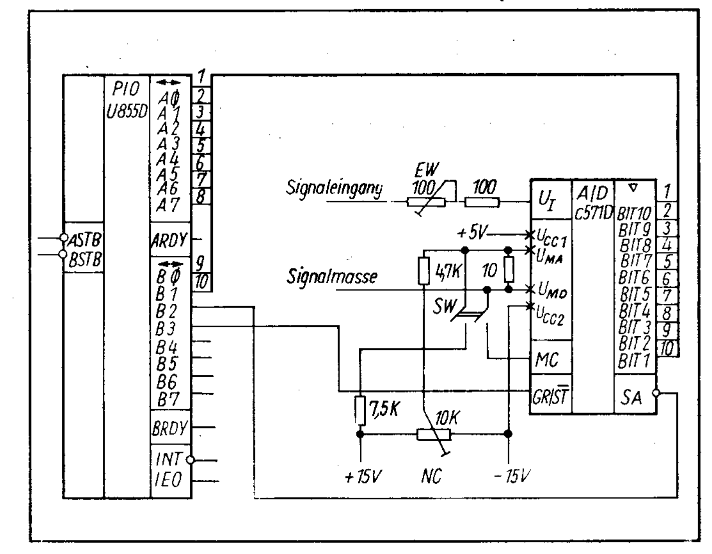


Bild 2 C 571 D an PIO im Bitbetrieb



Dipl.-Ing. Christian Heß (27) studierte nach seiner Berufsausbildung mit Abitur seit 1981 an der Technischen Hochschule Leipzig in der Sektion Automatisierungsanlagen und diplomierte zur Automation des Elektronenspinresonanzspektrometers ERS 231.
1985 nahm er an der gleichen Einrichtung ein Forschungsstudium auf.
Er beschäftigt sich gegenwärtig mit Untersuchungen zur on-line-Spectrophotometrie.

Übernahme durch die CPU zur Verfügung. Aufgrund der Impulsverzögerung durch die Gatterlaufzeiten und durch C1 wird außerdem die Herstellerforderung nach einer Verzögerungszeit $t_v < 500$ ns, bis die Daten an den Wandlerausgängen stabil sind /3/, erfüllt.

Software

Nach der PIO-Initialisierung auf Byte-Eingabe (Mode 1) wird ein CTC-Kanal (z. B. Kanal 3) interruptfähig als Zähler bis 1 programmiert.

Nach Festlegung der ISR-Adresse wird der Interrupt erlaubt (EI), und die CPU tritt in den Halt-Zustand. Sendet das externe Gerät sein SYN-Signal, wird die A/D-Wandlung ausgelöst, der Zähler dekrementiert und damit ein Interrupt an die CPU gemeldet. In der dadurch ausgelösten /SR wird zunächst lediglich eine neue ISR-Adresse festgelegt; gleichzeitig setzt der ADU seinen ersten Wert um und speichert ihn in den PIO-Eingaberegistern ab. Nach RETI wird das Programm einen Befehl nach dem HALT fortgesetzt. Dort könnte ein Sprung auf den vor dem HALT stehenden Befehl EI erfolgen. Jetzt kann die zyklische Verarbeitung des (n-1)-ten Meßwertes während der Umsetzung des n-ten Wertes erfolgen.

Für den C 571 D sind zwei IN-Befehle notwendig, um die 10 Bit in die CPU zu übernehmen. Es ist softwaremäßig zu sichern, daß zwischen beiden IN-Befehlen kein Strobeimpuls auftritt.

Wird keine Synchronisation durch externe Geräte benötigt, bleibt SYN unbeschaltet, bzw. die Gatter D 1.1 und D 2.1 können ganz entfallen. Der Ablauf beginnt dann mit /HALT = L. Sollen externe Geräte im Zeitablauf zyklisch vom Mikrorechner gesteuert und mit der A/D-Wandlung synchronisiert werden, so wird ein CTC-Kanal im Zeitgebermode interruptfähig initialisiert. Die Leitung SYN wird außer mit D 2.1 mit diesem Zeitgeberkanal verbunden (in Bild 3 gestrichelt). Die Verbindung vom Ausgang von D 2.1 zum Zähler entfällt. Der Zeitgeberkanal sendet bei jedem Übergang des Rückwärtszählers einen Impuls aus, der die externen Geräte synchronisiert und die A/D-Wandlung auslöst. Gleichzeitig wird ein Interrupt an die CPU gemeldet. Die Software verändert sich außer in der Phase der CTC-Initialisierung nicht.

Einem erhöhten Hardwareaufwand von zwei Logikschaltkreisen und der Benutzung eines CTC-Zähler- oder Zeitgeberkanals steht gegenüber, daß sich die Meßzeit t_M um die Umsetzzeit t_c sowie um die Programmlaufzeiten für den ADU-Start und die Statusauswertung verkürzt. Hinzu kommt die Zeit für die Ausführung des Interrupts und für die Rückkehr vom Interrupt.

Die Meßzeit ergibt sich somit nach Gl. (2).

$$t_M = t_{LESEN} + t_{INT} = 33\,\mu s \qquad (2)$$

t_{INT} Zeit für Ausführen des Interrupts, für RETI und für Relativsprung auf EI und HALT (53 Takte)

PIO im Byte-Eingabe-Mode und selbständige Umsetzung durch den C 571 D

Zur weiteren Verringerung der Meßzeit t_M wurde eine Schaltung nach Bild 5 realisiert, bei der auf die Nutzung des Interruptsystems verzichtet wurde. Das zugehörige Zeitverhalten ist in Bild 6 dargestellt.

Nach einem Initialstart des ADU werden die weiteren Umsetzungen durch seine eigenen Statusmeldungen ausgelöst. Gleichzeitig werden die Daten in die PIO-Eingaberegister eingeschrieben. Die CPU kann somit zu jedem beliebigen Zeitpunkt den zuletzt umgesetzten Wert von der PIO übernehmen. Dazu wurde durch eine Strobesperre hardwaremäßig realisiert, daß während und zwischen den beiden notwendigen IN-Befehlen zum Einlesen der 10 Bit aus der PIO in die CPU kein Strobe möglich ist.

Ablauf

Mit einem aktiven Resetsignal /RES wird das RS-Flipflop gesetzt und über /SA mit der erforderlichen Verzögerung rückgesetzt. Dadurch werden die Datenausgänge des ADU hochohmig geschaltet und die Umsetzung gestartet. Nach der Umsetzung werden /SA = L und mit der erforderlichen Verzögerung (siehe Bild 4) SA' = H. Bei nichtgesperrtem Strobe (Q3 = H) gelangt die Statusmeldung, durch die RS-Flipflops 1 und 4 auf eine Dauer von ca. 200 ns verkürzt, an die Strobeeingänge /ASTB und /BSTB der PIO. Damit werden die Daten in die Eingaberegister der PIO eingeschrieben, und gleichzeitig wird durch die Beschaltung von /STB an D 1.1, verzögert durch C4, eine neue Umsetzung nach dem

Bild 4 Zeitverhalten für Schaltung nach Bild 3

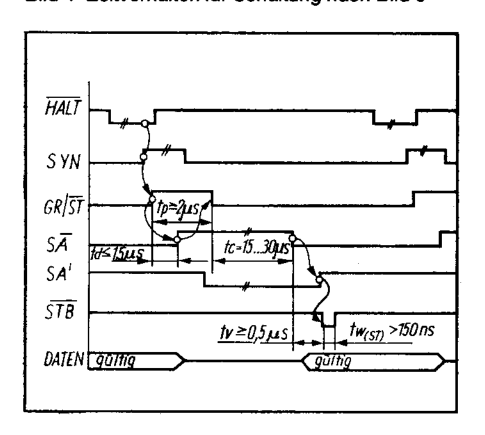


Bild 6 Zeitverhalten für Schaltung nach Bild 5

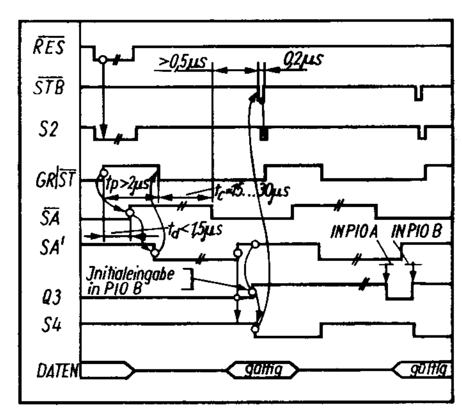


Bild 3 Byteeingabe mit Auslösung der A/D-Wandlung durch /HALT & SYN

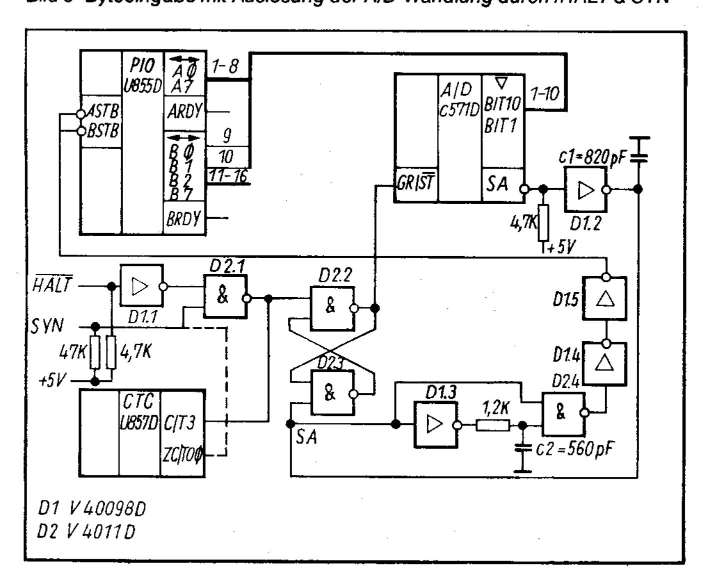
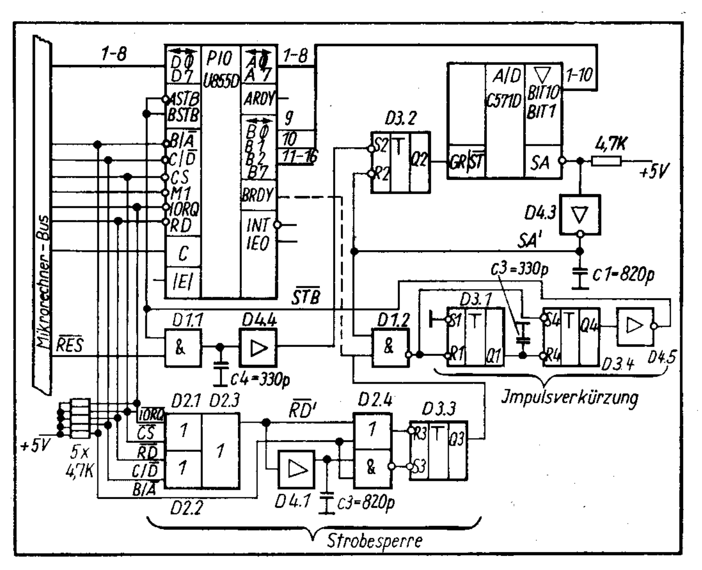


Bild 5 Selbständige Umsetzung durch den C 571 D



Strobeimpuls ausgelöst. Die ersten gültigen Daten werden eingeschrieben, wenn /RES H ist, bevor SA' = H wird, da sonst die Umsetzung sofort wieder ausgelöst wird und beim Strobeimpuls u. U. die Datenleitungen bereits wieder hochohmig sind. Aufgrund der üblichen Realisierungen des /RES stellt dies aber keine Einschränkung der Funktionsfähigkeit dar, und auf eine Impulsverkürzungsschaltung hierfür kann verzichtet werden.

Strobesperre

Zum Einlesen der im PIO-Eingaberegister abgespeicherten Daten eignet sich z. B. folgende Befehlsfolge:

```
IN  PIOA
LD  E, A
IN  PIOB
LD  D, A
```

Durch die Strobesperre wird gewährleistet, daß während und zwischen den beiden eigentlichen Portleseoperationen (Takt-Zustände T3 der Maschinenzyklen M2 der CPU-IN-Befehle) das RS-Flipflop 3 rückgesetzt ist und der Eingang der L-aktiven Impulsverkürzungsschaltung H-Potential führt. Ein Strobeimpuls wird somit verzögert, bis Q3 gesetzt wird.
Zur Erkennung einer bevorstehenden Leseoperation der PIO durch die CPU dient das L-aktive Signal
/RD' = /IORQ + /CS + /RD + C/$\overline{D}$.
Die Rücksetzung /R3 = /RD' + B/$\overline{A}$ wird nur im Befehl IN PIOA erfüllt. Q3 wird auf L geschaltet und damit ein Strobe so lange verzögert, bis /S3 = $\overline{/RD' \cdot B/\overline{A}}$ durch den Befehl IN PIOB das RS-Flipflop wieder setzt und Strobe erlaubt wird. Die Impulsverkürzung wurde aus zwei RS-Flipflops realisiert, da das NAND-Gatter D 1.4 in einer anderen Schaltung verwendet wurde und auf einen zusätzlichen Schaltkreis verzichtet werden sollte. Die CPU sendet die Signale /IORQ, /CS, /RD, C/$\overline{D}$ und B/$\overline{A}$ mindestens 2 Takte (ca. 800 ns) vor dem eigentlichen Lesen der Daten /4/. Dadurch ist gewährleistet, daß unter Berücksichtigung der Verzögerungs- und Flankenübergangszeiten das Rücksetzen des RS-Flipflops 3 und, im worst case, ein unmittelbar vor der Bedingung /R3 = L begonnener Strobeimpuls (Dauer ca. 200 ns) beendet sind. Die Gesamtverzögerungszeit der Strobesperrung (realisiert mit CMOS-Schaltkreisen bei 5 V Betriebsspannung) beträgt in der realisierten Variante 200 ns, die Gesamtflankenübergangszeit 60 ns.
Der Hersteller garantiert aber nur, daß die Gesamtverzögerungszeit $t_{vges.} \leqq 1{,}2\mu s$ ist. Für industrielle Anwendungen empfiehlt sich deshalb der Einsatz von LS-OR-Gattern des DL 032 D. Bei der Strobefreigabe durch Setzen des RS-Flipflops 3 ist es notwendig, /S3 zu verzögern, bis die Portleseoperation im Befehl IN PIOB erfolgt ist. Das wird durch C3 realisiert. In der Initialisierungsphase der PIO erfolgt durch eine Initialeingabe in Port B ein Anfangssetzen des RS-Flipflops.
Eine einfachere Realisierung der Strobesperre stellt die Beschaltung von BRDY der PIO an D 1.2 dar (gestrichelt dargestellt). Die Verbindung mit Q3 entfällt, wodurch D2 eingespart werden kann. In dieser Variante wird nach einem Strobeimpuls noch eine Umsetzung ausgelöst, der nächste Strobe und somit der nächste Start einer Umsetzung werden aber unterbunden, bis PIO-Port B gelesen wurde.

Synchronisation der A/D-Wandlung

Zur Synchronisation der A/D-Wandlung durch ein externes /SYN-Signal wird dieses an beide Eingänge von D 1.1 geschaltet. Die Verbindungen mit /STB und /RES entfallen. Durch C4 kann die Auslösung der Umsetzung verzögert werden, bis evtl. auftretende Übergangsvorgänge des Analogsignals (z. B. durch Choppern oder Multiplexen) abgeklungen sind.
Durch den Einsatz von 3 bzw. 4 Logikschaltkreisen kann die Zeit, bis der zuletzt umgesetzte ADU-Wert in einem Doppelregister der CPU abgespeichert ist, auf eine Meßzeit nach Gl. (3) verringert werden.

$$t_M = t_{LESEN} = 12\mu s \qquad (3)$$

Verwendung des C 570 D

Bei Verwendung des C 570 D kann ggf. die Abgleichschaltung nach Bild 2 entfallen. (Siehe hierzu /1/.) Die Bits 10 und 9 entfallen, so daß alle Daten über PIO-Port A gelesen werden können. In den Bildern 3 und 5 bleiben somit die PIO-Ports B unbenutzt. Weiterhin können in Bild 5 die Gatter D 4.1, D 1.3 und D 3.3 sowie C3 entfallen. Der Ausgang von D 2.4 wird direkt mit D 1.2 verbunden. Bei Strobesperrung durch RDY der PIO ist ARDY zu verwenden.
Durch die Softwarereduzierung kann z. B. die Lesezeit auf 4,5 μs gesenkt werden, bis die Daten dem Akkumulator zur Verfügung stehen.

Schlußbemerkung

Durch die Realisierung von Hardwarestrukturen bei den beschriebenen Applikationen unter Verwendung von, bei den auftretenden Frequenzen noch als verlustarm zu bezeichnenden, CMOS-Schaltkreisen kann die Meßzeit bis auf 12 μs (C 570 D: 4,5 μs) gesenkt werden.

Bei Umsetzung nach Bild 3 unter Verwendung dynamischer RAMs im verwendeten Mikrorechner und deren Refresh im Transparent-Mode ist längeres Verweilen im Haltzustand der CPU zu vermeiden, da aufgrund zu häufigen Refreshs thermische Probleme auftreten könnten.
Bezüglich der aufgenommenen bzw. aufgebrachten Ausgangsströme des verwendeten CMOS-Gatters beim Treiben von GR/$\overline{ST}$ treten keine Probleme auf, da laut /7/ der Steuerstrom $/I_{I11}/ \leq 40\mu A$ ist.
Die Umsetzzeit des eingesetzten Exemplars betrug $t_c = 18\mu s$. Die vorgestellten Applikationen stellen Beispiele dar, die Anregungen für Schaltungen mit dem C 571 D an U880-Systemen geben sollen.

Literatur

/1/ Krüger, K.: Analogwerteingabe in Kleincomputer. Radio Ferns. Elektron., Berlin 35 (1986) 9, S. 557–558
/2/ Data Acquisition Databook: 10 Bit Analog to Digital Converter AD 571. Analog Devices 1982
/3/ Datenblatt: C 570 D, C 571 D. VEB Halbleiterwerk Frankfurt/O.
/4/ Kieser, H.; Meder, M.: Mikroprozessortechnik. VEB Verlag Technik, Berlin 1984
/5/ Schmidt, T.; Greiner, T.: Zeitoptimierte A/D-D/A-Baugruppe mit C 571/C 565. Mikroprozessortechnik, Berlin 1 (1987) 9, S. 262
/6/ Katalog: Aktive elektronische Bauelemente. VEB Kombinat Mikroelektronik, Erfurt 1985
/7/ TGL 43269. 10-Bit-Analog-Digital-Wandler-Schaltkreise C 571 D.

✉ KONTAKT ☎

Technische Hochschule Leipzig,
Sektion Automatisierungsanlagen,
WB 3, Karl-Liebknecht-Str. 132, Leipzig, 7030;
Tel. 3943139

Einplatinenrechner als Schnittstelle zwischen Analog- und Digitaltechnik

Dieter Zühlke
VEB Mikroelektronik „Karl Marx" Erfurt

Vorgestellt wird eine Platine, die bei Klein- und Bürorechnern als Bindeglied zu Steuerungs- und Regelungsprozessen sowie für die Erfassung und Abspeicherung von Analogsignalen gedacht ist. Das System umfaßt jeweils 8 Multiplexkanäle bei AD-Wandlung (C 571) und DA-Wandlung (C 565) sowie einen Einchip-Mikrorechner (EMR) U 884 mit SRAM (z. B. U 6516) als Speicherblock und drei verschiedene digitale Anschlußmöglichkeiten.

Vorbemerkungen

Die ADDA-EMR 02 und 03 ist eine universelle Platine (K1520-Format), die für keinen speziellen Anwendungsfall entwickelt wurde. Im folgenden sollen neben der Beschreibung der Funktionsblöcke auch einige Anwendungsbeispiele angedeutet werden. Hauptanwendungsgruppen sind neben dem Rationalisierungsmittelbau auch Schul- und Bildungseinrichtungen.

Die Funktionsblöcke (Bild 1) sind:

- EMR-Speicher-Block (U 882XM oder U 884XM) Variante 02 mit 7×28poligen Fas-

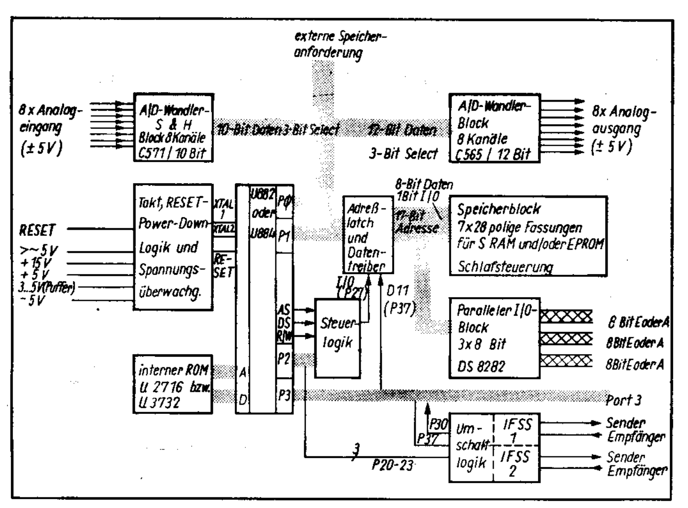


Bild 1 Funktionsblöcke der ADDA-EMR

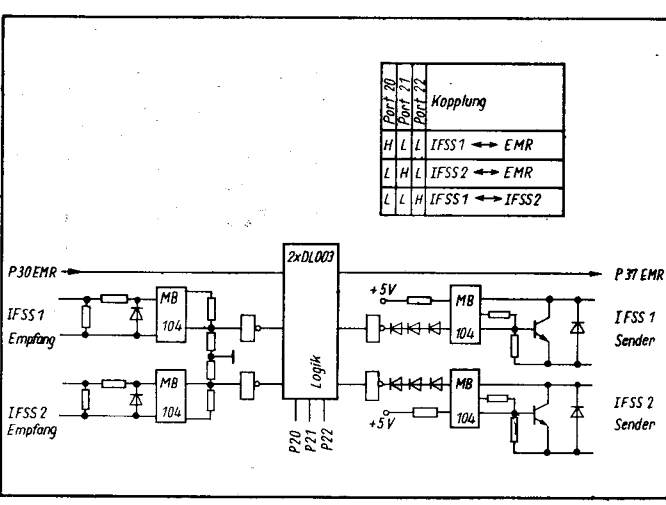


Bild 3 IFSS-Block

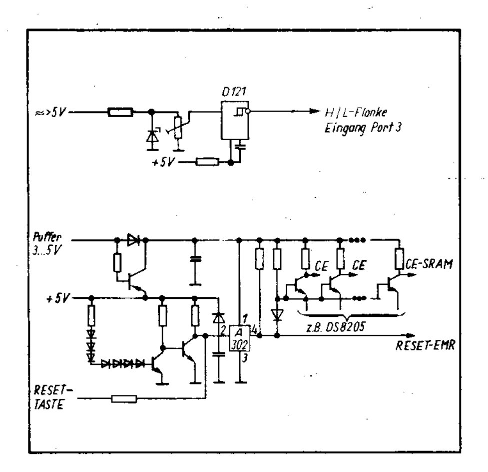


Bild 2 RESET- und Power-Down-Logik

sungen für SRAM, EPROM
Variante 03 mit 3×28poligen Fassungen für EPROM und 12 × U 224 als SRAM

- RESET- und Schlafsteuerungslogik
- IFSS-Block
- 3 × 8-Bit-Ein- oder Ausgabelatches
- DMA-Anschluß
- 10-Bit-ADW-Block mit Sample & Hold
- 12-Bit-DAW-Block.

Durch die Blockstruktur wird eine ökonomische Teilbestückung ermöglicht. Die Versorgungsspannungen betragen +/−15 V für den Analogteil und +5 V für den Digitalteil sowie beim Akku 3–5 V bei Power-Down-Betrieb.

EMR-Speicher-Block

Die Platine kann mit den EMR-Schaltkreisen U 8840M und EPROM U 2732C oder U 8841M (für Power-Down-Betrieb) und EPROM 02732C bzw. U 8820M und EPROM U 2716C oder U 8821M (für Power-Down-Betrieb) und EPROM U 2716C bestückt werden.

Durch Programmierung der Ports 0 und 1 auf Adreß-Datenport wird über ein 16-Bit-Adreßlatch und einen 8-Bit-Datentreiber der Anschluß zwischen dem EMR und dem externen Speicher auf der Platine gewährleistet. Beim Speicher-Block besteht bei beiden Varianten die Möglichkeit, den EMR und die SRAMs durch einen Akku zu puffern. Eine Trennung von Programm- und Datenspeicherbereich ist durch Anschluß P34 (Programmierung DM) möglich, und die Schreibsperren für die SRAMs sind auf der Leiterplatte einschaltbar.
Besonders die ADDA-EMR 02 ist hinsichtlich der Weiterentwicklung der Speicherschaltkreise in ihrer Bestückung offen.

RESET- und Schlafsteuerungslogik

Diese beiden Funktionen, verbunden mit einer Spannungsüberwachung, sind im Bild 2 dargestellt. Die Überwachungsschaltung wertet die positiven Halbwellen einer Wechselspannung (> 5 V) aus und löst beim Unterschreiten einer Triggerschwelle eine H/L-Flanke am Ausgang aus, die vom EMR als Interruptquelle genutzt werden kann. Auf die Beschreibung der RESET- und Power-Down-Logik soll hier verzichtet werden.

Ihre Aufgaben sind:

– Sicheres Umschalten in den Schlafzustand der Register des EMR und der SRAMs
– exakte Rücknahme des aktiven RESET-Zustandes auch bei langsam steigender Betriebsspannung.

IFSS-Block

Als Basis der Kopplung dient der IFSS-Standardanschluß (Interface zum sternförmigen Anschluß von Ein- und Ausgabegeräten mit serieller Informationsübertragung). Dieser besteht aus zwei Stromschleifen (Sender und Empfänger) für 20 mA. Über Optokoppler erfolgt eine galvanische Trennung zwischen der ADDA-EMR und der Übertragungsleitung. Der EMR bestimmt die Bitrate (110 ... 19 200) und das serielle Datenformat der asynchronen Übertragung. Durch die Anschlüsse P20, P21, P22 kann der EMR über eine Verknüpfungslogik entscheiden, mit welchen der 2 Geräte (z. B. aktive Tastatur und Wirtsrechner) er in Verbindung tritt oder ob er die Datenleitung durchschaltet (Bild 3).

3 × 8-Bit-Ein- oder Ausgabelatch

Als Latch sind DS8282D verwendet worden, die je nach Bestückung als Ein- oder Ausgabetor fungieren. Die Datenleitungen sind an die getriebenen Speicherleitungen angeschlossen. Über die Adreßleitung A8, A9 und A10 werden die Latches angesprochen. Durch den Anschluß P27 (L → I/U und H → Speicher) wird ohne Einschränkung des Daten- und Programmspeicherbereiches eine klare Trennung von diesen erreicht. Dies ist besonders bei der Abspeicherung umfangreicher Meßreihen und bei dem externen Speicherzugriff von Vorteil.

DMA-Anschluß

Um einen schnellen und damit auch umfangreichen Datentransfer zu ermöglichen, ist ein direkter Speicherzugriff notwendig. Dies wird vom EMR programmtechnisch durch die Möglichkeit des hochohmig Schaltens von Port 0 und 1 und der Steuerleitungen AS, DS und R/W unterstützt. Die Schnittstelle umfaßt diese Leitungen zuzüglich der notwendigen Leitungen BUSAK (P32) und BUSRQ (P35). Wesentliches Kriterium der Gegenstelle ist, daß sie zeitmultiplex Adressen (16 Bit) und Daten (8 Bit) senden und empfangen kann.

ADW-Block

Im Bild 4 ist das Schaltbild des Analog-Digital-Wandlers dargestellt. Die Ports P10 ... 17 und P00 ... 03 sind auf Eingang programmiert. Über das obere Nibbel von Port 0 wird der Kanal des Multiplexers (V4051D) ausgewählt, der seinen Analogwert auf den Kondensator C in der Abtastphase gibt. Danach wandelt der C571D den Wert, und er kann vom EMR zur Weiterverarbeitung gelesen werden. Mit dem Anschluß P23 wird die Steuerlogik gestartet, und am Anschluß P24 wird das Ende der ADW vom C571D registriert. Im nächsten Abschnitt wird ein Programm für 8-Kanal-AD- und -DA-Wandlung gezeigt.
Technische Daten:

Anzahl der Kanäle:	maximal 8
Spannungspegel:	+/−5 V
Auflösung:	10 Bit bei C 571D 8 Bit bei C 570D
Umsetzzeit:	15 ... 30 µs bei C 571D (typisch 18 ... 20 µs) 15 ... 40 µs bei C 570D
Abtastzeit:	5 µs

DAW-Block

Der Digital-Analog-Wandler ist im Bild 5 dargestellt. Die Ports 0 und 1 sind auf Ausgabe

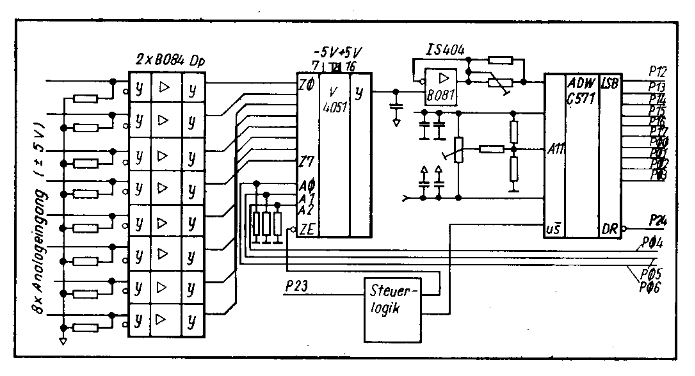


Bild 4 A/D-Wandler-Block

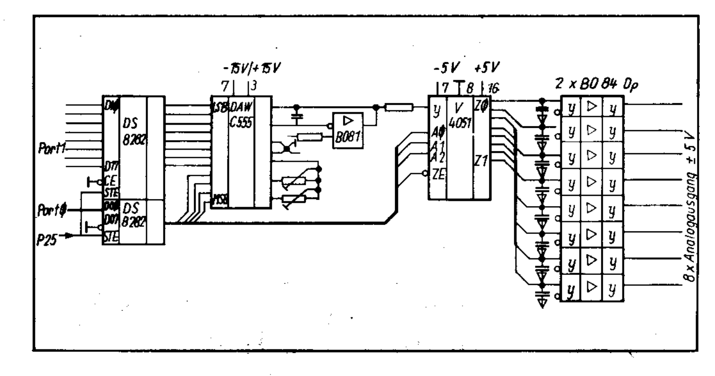


Bild 5 D/A-Wandler-Block

```
Z8ASM     3.03
LOC    OBJ CODE    STMT SOURCE STATEMENT

         ADDA_ANALOG          MODULE

          !Die Daten vom ADW werden ueber die Adressen im Reg. 12,13 gezeigert!
          !Die Daten fuer DAW werden ueber die Adressen im Reg. 14,15 gezeigert!
          !Die Anzahl der Daten werden im Register 11 uebergeben!

         CONSTANT

         PORT_0      :=   R0
         PORT_1      :=   R1
         PORT_2      :=   R2
         ZAEHLER     :=   R11
         H_DATEN     :=   R9
         L_DATEN     :=   R10
         ZIELADR     :=   RR12
         H_ZIELADR   :=   R12
         L_ZIELADR   :=   R13
         QUELLADR    :=   RR14
         H_QUELLADR  :=   R14
         L_QUELLADR  :=   R15


          !P23 AD-Start H/L-Flanke!
          !P24 STS vom ADW (L-aktiv)!
          !P25 DA-Start H-aktiv!
          !P10-P17,P00-P03 12Bit Ausgang bzw. Eingang (LSB ==> P10)!
          !P04-P07 8 Kanal Ausgang!
          !H_ZIELADR fuer ADW (H-Teil)!
          !L_ZIELADR fuer ADW (L-Teil)!
          !H_QUELLADR fuer DAW (H-Teil)!
          !L_QUELLADR fuer DAW (L-Teil)!


         GLOBAL

                     *REL

P 0000   ANALOG      PROCEDURE

         ENTRY

P 0000 70 FD                   PUSH   RP
P 0002 31 00                   SRP    #0
P 0004 82 9E         SCHLEIFE: LDE    H_DATEN,@QUELLADR
P 0006 A0 EE                   INCW   QUELLADR
P 0008 82 AE                   LDE    L_DATEN,@QUELLADR
P 000A E6 F8 04                LD     P01M,#%(2)00000100
P 000D 08 E9                   LD     PORT_0,H_DATEN
P 000F 56 E2 F7                AND    PORT_2,#%(2)11110111    !AD-Start!
P 0012 18 EA                   LD     PORT_1,L_DATEN
P 0014 A0 EE                   INCW   QUELLADR
P 0016 46 E2 20                OR     PORT_2,#%(2)00100000    !DA-Uebernahme!
P 0019 56 E2 DF                AND    PORT_2,#%(2)11011111    !DA-Uebernahme beendet!
P 001C E6 F8 0D                LD     P01M,#%(2)00001101      !Bereit fuer AD-Wert!
P 001F 66 E2 10      WARTEN:   TCM    PORT_2,#%(2)00010000
P 0022 6B FB                   JR     Z,WARTEN
P 0024 98 E0                   LD     H_DATEN,PORT_0            !H-Teil vom ADW!
P 0026 A8 E1                   LD     L_DATEN,PORT_1            !L-Teil vom ADW!
P 0028 46 E2 08                OR     PORT_2,#%(2)00001000      !AD-Start!
P 002B E6 F8 96                LD     P01M,#%(2)10010110
P 002E 92 9C                   LDE    @ZIELADR,H_DATEN
P 0030 A0 EC                   INCW   ZIELADR
P 0032 92 AC                   LDE    @ZIELADR,L_DATEN
P 0034 A0 EC                   INCW   ZIELADR
P 0036 BA CC                   DJNZ   ZAEHLER,SCHLEIFE
P 0038 50 FD                   POP    RP
P 003A AF                      RET

P 003B             END   ANALOG

                   END   ADDA_ANALOG

  0 errors
Assembly complete
%
```

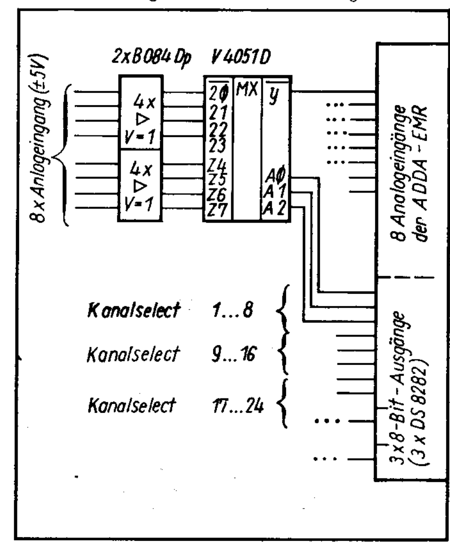
Bild 6 Beispiel für Treiberroutine

programmiert. Die 2 Latches (DS8282D) übernehmen mit der H/L-Flanke des Anschlusses P25 den aktuellen Digitalwert, den dann der C565D wandelt. Bei mehr als einem Kanal wird dieser Analogwert mittels eines Demultiplexers auf einem Kondensator gespeichert. Eine Programmroutine zur Auffrischung der gespeicherten Werte ist notwendig. Diese Routine kann man nun gut mit in den Programmablauf des ADW (Umsetzzeit) integrieren. Ein Beispiel für eine Treiberroutine (Bild 6) soll das veranschaulichen.

Technische Daten:

Anzahl der Kanäle:	maximal 8
Spannungspegel:	+/−5 V
Auflösung:	12 Bit bei C 565D
	10 Bit bei C 5650D
	8 Bit bei C5658D
Umsetzzeit:	im Beispiel durch ADW bestimmt!

Bild 8 Erhöhung der Anzahl der Analogkanäle



$R \approx \frac{+u_B - 5V}{I_{Zmax}}$ I_{Zmax} - Max. zulässiger Z-Strom

R

z.B. Ausgang 0 V

Analogeingang (±5V)

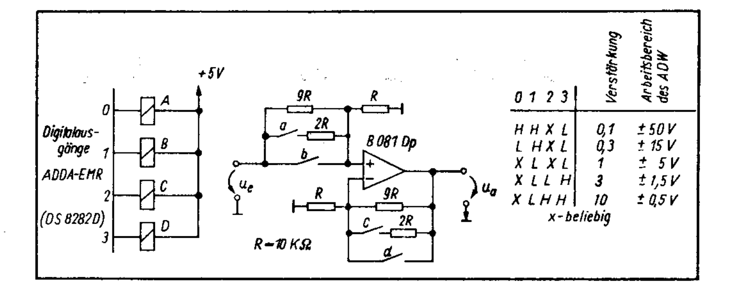
Re > 2000 · R bei 10 Bit Auflösung

I_{Dmax}

2 x SZ x 21/5.1 o.a.

Bild 7 Eingangsschutzschaltung

Bild 9 Automatische Meßbereichsumschaltung für A/D-Wandler



Anwendungshinweise

Die Analogeingänge sollten auf eine Spannung +/−5 V begrenzt werden (Bild 7) oder der vorhergehende OV sollte diese Betriebsspannungen haben.
An den Anschlüssen P30 bis P33 kann mit einer H/L-Flanke ein Interrupt ausgelöst und in die Analogwertaufnahme eingebunden werden. Dadurch ist es möglich,

– ein ständiges Abspeichern (Überschreiben) sofort oder zeitverzögert zu beenden (z. B. Havariefall)
– eine begrenzte Meßreihe vom Startimpuls an aufzunehmen, wobei die Zeitabstände zwischen den einzelnen Meßwerten von der AD-Programmroutine (100 μs ... Stunden) abhängen.

Bei Verwendung eines 7,37280-MHz-Quarzes ist eine uhrzeitsynchrone Arbeit des EMR möglich.
Wenn Meßwerte über sehr lange Zeiträume aufgenommen oder ausgegeben werden sollen, ist eine zeitgestaffelte Bearbeitung möglich (e-Funktionen).
Soll auf einfache Art die gespeicherte Meßwertreihe kontrolliert oder mit einer zweiten Meßreihe verglichen werden, so wird dies durch Anschluß der ADDA-EMR an einen Zweistrahloszillograf und durch die periodische Ausgabe dieser Meßreihen über zwei DA-Kanäle erreicht.
Der DA-Wandler-Block kann mit dem Speicherbereich auch als freiprogrammierbarer Funktionsgenerator verstanden werden, so zur Erzeugung von variablen Phasen- und Frequenzgängen für simultane Drehströme.
Eine Erhöhung der analogen Eingänge bis auf 64 Kanäle ist durch den Anschluß von V4051D und B084Dp nach Bild 8 möglich. Zu beachten ist, daß sich dann die Zeitpunkte für die Abtastung eines Kanals mit wachsender Signalzahl im gleichen Verhältnis vergrößern und sich die Offsets der in Reihe geschalteten OV multiplizieren. Ähnliches gilt für die Erhöhung der analogen Ausgänge.
Durch die Schaltung nach Bild 9 kann für A/D-Wandler eine automatische Meßbereichsumschaltung realisiert werden.

Schlußbemerkungen

Die Variabilität des Speicherblocks wurde durch eine hohe Anzahl an Wickelverbindungen bei der ADDA-EMR 02 ermöglicht.
Die Trennung von Speicher und I/O-Schnittstelle durch den Anschluß P27 erfordert einen erhöhten Programmaufwand bei der Nutzung dieser Schnittstelle.
Eine im Dauerbetrieb arbeitende Platine mit zusammengeschalteten analogen Ein- und Ausgängen arbeitet problemlos.
Der VEB Mikroelektronik „Karl Marx" Erfurt bietet ein Nachnutzungspaket mit begrenzter Anzahl durchkontaktierter Leiterkarten an.

Literatur

/1/ Technische Beschreibung: Einchip-Mikrorechner-Schaltkreis U881/U882. VEB Mikroelektronik „Karl Marx" Erfurt
/2/ Mikroelektronik-Information C 5650D, C565D
/3/ Mikroelektronik-Information C 570D, C 571D
/4/ Dietrich: 10- und 8-bit-AD-Wandlerschaltkreise C 571 und C 570. 11. Mikroelektronik-Bauelemente-Symposium Frankfurt (O.), 1985, S. 341–348
/5/ Kulesch; 8-, 10- und 12-Bit-DA-Wandlerschaltkreise. 11. Mikroelektronik-Bauelemente-Symposium Frankfurt (O.), 1985, S. 349–358

✉ KONTAKT ☎

VEB Mikroelektronik „Karl Marx" Erfurt,
Abt. CKM3, Rudolfstraße 47, Erfurt, 5010;
Tel. 51771 App. 148

Transformationsprogramm für dBaseII- und TURBO-PASCAL-Dateien

Dirk Mehlhorn, Hans-Jürgen Bauer
Technische Hochschule Leipzig,
Sektion Elektroenergieanlagen

Das Programm S1 realisiert die Transformation einer dBaseII-(bzw. REDABAS-) Datei in eine TURBO-PASCAL-(bzw. PASCAL880/S-) Direktzugriffsdatei oder die Rücktransformation einer TURBO-PASCAL-Direktzugriffsdatei in die dBaseII-Datei (oder in eine Datei gleicher Struktur). Beide Operationen können zu beliebiger Zeit ausgeführt werden. Bedingung ist, daß vor der Rücktransformation mindestens eine Transformation erfolgte.
Zusätzlich wird (nur bei der 1. Transformation) eine TURBO-PASCAL-Quelltextdatei angelegt, die den Deklarationsteil für die erzeugte Direktzugriffsdatei enthält.

Aufruf

S1 [[LW:] [DATEINAM. TYP] [0]
{0 für Output (Transformation dBASEII ⇒ TURBO-PASCAL)}
S1 [[LW:]DATEINAM. TYP] [1]
{I für Input (Transformation TURBO-PASCAL ⇒ dBase II)}
(DATEINAM. TYP: = Name/Typ der dBaseII- bzw. REDABAS-Datei)

Leistung

1. Output
S1 erzeugt
1. Die TURBO-PASCAL-Direktzugriffsdatei DATEINAM. RAN auf dem (angegebenen/aktuellen) Laufwerk.
2. Die Deklarations-Quelltextdatei DATEINAM. DEK auf dem (angegebenen/aktuellen) Laufwerk.
(Extensionen werden automatisch gesetzt).

2. Input
S1 überträgt den Inhalt der TURBO-PASCAL-Direktzugriffsdatei DATEINAM. RAN in die dBase II-Datei DATEINAM. TYP auf dem (angegebenen/aktuellen) Laufwerk.

Parameter
maximale Breite der dBaseII-Datei:
1000 Zeichen/Zeile
maximale Länge der dBaseII-Datei:
65535 Zeilen

Transformationsparameter
Standard:
Felder in dBaseII ⇒ Variablen in TURBO-PASCAL

TYP	LÄNGE	DEZIMALST.	TYP [LÄNGE]
C	n	0	STRING [n]
L	1	0	BOOLEAN
N	1–2	0	BYTE
N	3–4	0	INTEGER
N	5-max	0	REAL
N	2-max	1–68	REAL

Weitere Liefermöglichkeiten:

Felder in dBaseII ⇒	Variablen in TURBO-PASCAL
N |1−m{m≤4}|	BYTE
B |m+1| bis 7	INTEGER

Transformationszeit

Die Transformationszeit wird durch die Lese-Schreibgeschwindigkeit der Laufwerke und den Umfang der Datei bestimmt. Eine Transformation dauert etwa solange wie ein Kopieren der Datei von Diskette zu Diskette mittels PIP oder POWER.

Speicherplatzbedarf auf Diskette

Deklarations-Quelltextdatei: 2 KByte
TURBO-PASCAL-Direktzugriffsdatei: entsprechend dBaseII-Quelle.
(Anmerkung: In den meisten Fällen wird weniger Speicherplatz benötigt, da numerische Felder in dBaseII in die internen TURBO-PASCAL-Formate real/integer/byte transformiert werden.)

UPDATE

1. Output:
Die Direktzugriffsdatei DATEINAM.RAN wird bei wiederholtem Aufruf von S1 jedesmal überschrieben.
S1 prüft bei jedem Aufruf, ob die Datei DATEINAM. DEK auf LW vorhanden ist und legt sie im Bedarfsfall an.
2. Input:
Die dBaseII-Datei wird mit den Daten der PASCAL-Direktzugriffsdatei DATEINAM. RAN aktualisiert, alle in PASCAL vorgenommenen Änderungen (z. B. Löschen, Anfügen, Ändern der Sätze der Direktzugriffsdatei) werden in die dBaseII-Datei übernommen.

Einsatzmöglichkeiten

Für dBaseII-Programmierer:
Einbindung von TURBO-PASCAL-Programmen (z. B. für umfangreiche numerische Berechnungen) in dBaseII-Programme.
Für TURBO-PASCAL-Programmierer:
Nutzung der von dBaseII angebotenen Mög-

```
program s1bsp;
{*I s1bsp.dek}
begin
assign(s1bspfile,'s1bsp.ran');
reset(s1bspfile);
{Kontrollausgabe}
while not eof(s1bspfile) do
begin
  clrscr;
  read(s1bspfile,s1bspsatvar);
  with s1bspsatvar do
  begin
    writeln('Kontrollausgabe S1BSP.RAN');
    writeln;
    writeln('AAAAAAAAAA=',aaaaaaaaaa);
    writeln('B=',b,' ; C=',c,' ; D=',d);
    writeln('E=',e,' ; F=',f);
    writeln('G=',g);
  end;
  writeln('weiter mit bel. Tastendruck');repeat until keypressed;
end;
{Anfuegen eines neuen Satzes}
seek(s1bspfile,filesize(s1bspfile));
s1bspsatvar.aaaaaaaaaa:='HALLO';
write(s1bspfile,s1bspsatvar);
close(s1bspfile);
end.
```

Bild 1 Beispielprogramm

lichkeiten der Dateneingabe/-Aufbereitung/-Plausibilitätsprüfung.
Zur Pflege und Wartung von Stammdaten mittels dBaseII.
Nutzung des Reportgenerators/Maskengenerators (ZIP) in dBaseII zur Datenausgabe.
Nutzung der Kompatibilität von dBaseII/REDABAS zu anderen Programmiersprachen (BASIC/FORTRAN), zu anderen Serviceprogrammen (Kalkulations-/Statistikprogramme, z. B. SUPERCALC/ABSTAT) sowie zum Grafikprogramm PCGRAF.
Hinweis:
S1 nutzt die SCP-Kommandozeile (ab 80 hex) zur Übergabe des Dateinamens der dBaseII-Datei und der Output-/Input-Option.
Einbindung von S1 in andere Programme:
1. dBaseII
– Output
Quit to ' [LW1:]S1 [LW2:]DATEINAM. TYP 0'
– Input
Quit to' [LW;:]S1 [LW2:]DATEINAM. TYP I'
(LW1 = Laufwerk, auf dem sich S1 befindet; LW2 = Laufwerk, auf dem sich die dBaseII-Datei DATEINAM. TYP befindet)

2. Zum Aufruf von S1 in mit TURBO-PASCAL erstellten Programmen wird die Procedur EX bereitgestellt
– Output/Input
mit CHAIN – AUFRUF
ex ('[LW:]S1.CHN', '[LW:]DATEINAM. TYP', '0'); mit EXECUTE-AUFRUF
ex('[LW:]S1. COM','[LW:] DATEINAM. TYP', '0'); (bzw. 'I')

Hinweis:
– Im rufenden Programm muß das Include-File EX.INC eingebunden sein ({○IEX.INC}-Anweisung).
– Für die Einbindung mittels CHAIN wird das Programm S1.CHN (S1.COM ohne TURBO-PASCAL-Laufzeitbibliothek) benötigt.

Beispiel
Im Beispiel werden benötigt:
LW A: dBase.COM
S1.COM
LW B: S1BSP.DBF
S1BSP.COM
Es werden folgende Dateien erzeugt:
LW B: S1BSP.DEK
S1BSP.RAN

Folgende dBaseII-Datei wurde in dBaseII mittels create erzeugt und durch append mit Daten gefüllt:
: display strukture
STRUKTURE FÜR FILE B: S1BSP.DBF
NUMBER OF RECORDS 00002
DATE OF LAST UPDATE 02/05/87
PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	AAAAAAAAAA	C	005	
002	B	N	002	
003	C	N	003	
004	D	N	004	
005	E	N	005	
006	F	N	008	004
007	G	L	001	
TOTAL				00029

: list
00001 aaaaa 22 333 4444 55555 66 6666 t
00002 AAAAA 99 999 9999 99999 999 .9999
f
: quit
Nach dem Verlassen von dBaseII meldet sich das Betriebssystem:
A⟩
Von der Betriebssystemebene wird S1 wie folgt aufgerufen:
A⟩ S1 B:S1BSP.DBF 0⟨CR⟩
Es erfolgt die Transformation dBaseII ⇒ PASCAL.
Die folgende PASCAL-Deklarations-Datei S1BSP.DEK wird erstellt:
A⟩ TYPE B:S1BSP.DEK ⟨CR⟩

```
{Deklarationsteil für TURBO-PASCAL – Datei s1 bsp. RAN }
const
Spaltenzahl = 7;
type
s1bspsatz = record
AAAAAAAAAA    : string [5];
B             : byte;
C             : integer;
D             : integer;
E             : real;
{numerisches dBaseII-Feld E ist größer als 4, deshalb real }
F             : real;
G             : boolean;
              end;
var
s1bspfile     : file of s1bspsatz;
s1bspsatvar   : s1bspsatz;
```

Das TURBO-PASCAL Programm S1BSP.COM (Bild 1) gibt die TURBO-PASCAL-Quelltextdatei S1BSP.RAN aus und hängt einen weiteren Satz an das Ende der Datei an.
Aufruf des Programms S1BSP.COM von der Betriebssystemebene:
A⟩ B:S1BSP⟨CR⟩
Rücktransformation der mit S1BSP.COM veränderten Datei S1BSP.RAN in die dBaseII-Datei S1BSP.DBF durch folgenden Aufruf von der Betriebssystemebene:
A⟩ S1 B:BSP.DBF I ⟨CR⟩
Der Inhalt von S1BSP.DBF wird aktualisiert. In dBaseII kann (list oder display) die Kontrolle der mit einem Datenbestand der Datei S1BSP.DBF ausgeführten Manipulationen erfolgen.

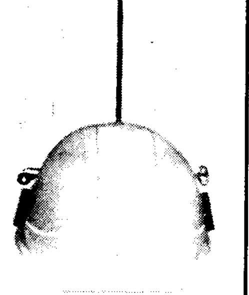
✉ KONTAKT ☎

Technische Hochschule Leipzig, Sektion Elektroenergieanlagen, Karl-Liebknecht-Str. 132, Leipzig, 7030; Tel. 3943187

Mikroelektronik und Computertechnik aus der Sicht der Karikaturisten

Auf der KARIGRAFIE'87 in Berlin – veranstaltet wurde sie von den Sektionen Pressezeichner/Karikaturisten des Verbandes Bildender Künstler der DDR und des Verbandes der Journalisten der DDR – entdeckt:





◀ *Ergonomischer Hammer mit digitaler Anzeige für totgeschlagene Zeit von Roland Jäger*

▶ *CAD/CAM-Helm von Rainer Schwalme*

PASCAL

(Teil 1)

Dr. Claus Kofer
Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden

0. Einführung

PASCAL wurde um 1970 von N. Wirth an der ETH Zürich als Lehrsprache für die Programmierausbildung entwickelt. Heute ist sie in der Welt eine der am meisten angewendeten Programmiersprachen. Dafür gibt es eine Reihe von Gründen: PASCAL ist universell anwendbar, hat eine klare Struktur, ist einfach zu erlernen, ermöglicht die effektive Compilierung und steht auf fast allen Rechnern in den wichtigsten Betriebssystemen zur Verfügung.
PASCAL-Programme sind in wesentlich geringerem Maße betriebssystem- und rechnerabhängig als Programme anderer Programmiersprachen. Sie gestatten damit die Entwicklung übertragbarer und langlebiger Softwarelösungen.
Seit 1983 gibt es das TURBO-PASCAL-System. Es wurde von der BORLAND Corp., USA, für den Einsatz auf Arbeitsplatz- und Personalcomputern entwickelt.
Auch in der DDR ist es für alle Büro- und Personalcomputer, die unter einem CP/M-kompatiblen Betriebssystem laufen, verfügbar.
Im Vergleich mit den bis 1983 bekannten PASCAL-Compilern besticht TURBO-PASCAL durch seine außerordentlich hohe Übersetzerleistung von ca. 1000 Zeilen Quellprogramm/Minute. Dies und ein integrierter Quelltexteditor führen zu einem sehr kurzen Programmentwicklungszyklus: Editieren, Compilieren, Testen. Die übersetzten PASCAL-Programme sind sehr kompakt, so daß ihre maximale Größe bei ca. 2000 Quellzeilen liegt. Der Sprachumfang von TURBO-PASCAL entspricht dem von Jensen/Wirth in „PASCAL: User Manual an Report" dargelegten Standard-PASCAL. Es gibt wenig Einschränkungen, aber viele nützliche Erweiterungen wie

- Datentyp STRING
- Direktzugriffsfiles
- Konstanten der strukturierten Datentypen
- Überlagerungsmechanismus für Prozeduren
- Auslösen von Betriebssystemrufen
- Aufruf von Maschinenkodeunterprogrammen
- Einfügen von Maschinenkodepassagen.

Die Grundlage für diesen Beitrag bildet der Themenplan einer Weiterbildungsveranstaltung zur Anwendung der Programmiersprache PASCAL, die der Autor seit mehreren Jahren für EDV-Anwender durchführt. Die einzelnen Elemente der Sprache werden aufeinander aufbauend vorgestellt und ihre Anwendung an Beispielen demonstriert.
Wo es notwendig ist, auf Details einzugehen, wird die Lösung von TURBO-PASCAL geschildert.
Darüber hinaus werden Einschränkungen und Besonderheiten dieses Systems bei der Behandlung der einzelnen PASCAL-Elemente besprochen.
Alle angegebenen Programmbeispiele laufen und sind mit TURBO-PASCAL auf PC 1715 getestet worden.
Den Abschluß bildet ein Kapitel, in dem Erweiterungen und Spezifika von TURBO-PASCAL vorgestellt werden.

1. Darstellung der Sprache

Ein Kurs zur Vermittlung einer Programmiersprache muß zwei Aufgaben lösen: Erstens sind darzustellen die vorhandenen Sprachelemente und die an sie geknüpften Inhalte, d. h. die Semantik, und zweitens in welcher Form die Sprachelemente korrekt niederzuschreiben und zu kombinieren sind, die Syntax. Während zur Darstellung der Syntax formale Beschreibungsverfahren angewendet werden, ist es immer noch üblich, die Semantik verbal darzulegen. Eine besonders prägnante Form der Syntaxbeschreibung sind Syntaxdiagramme. Sie wurden erstmals im Zusammenhang mit PASCAL in breitem Umfang genutzt. Auch in diesem Beitrag werden sie angewendet.
Das Bild 1.1 zeigt vier Grundformen von Syntaxdiagrammen. Bei der *Sequenz* (Bild 1.1a) wird a durch eine Folge von b und c gebildet. Während bei einer *Alternative* (Bild 1.1b) a entweder durch b oder durch c gebildet wird. Natürlich können Sequenz und Alternative um Elemente d, e, f, ... angereichert werden, falls es erforderlich ist. In einer *Liste* (Bilder 1.1c und 1.1d) wird a durch eine zunächst beliebig lange Folge von b gebildet. Während die Liste nach Bild 1.1c jedoch mindestens einmal b enthält, kann die nach Bild 1.1d gebildete Liste auch leer sein.
Die Anwendung der Grundformen wird im Bild 1.2 am Beispiel des Syntaxdiagramms für die Repeat-Anweisung gezeigt. Sie enthält im Inneren eine nicht leere Liste von Anweisungen, die voneinander durch das Semikolon getrennt sind. Die Sequenz von REPEAT, der Anweisungsliste, UNTIL und „ausdruck" bildet schließlich die komplette Repeat-Anweisung. Syntaktisch korrekte Formen entstehen durch eine Traversierung des Syntaxdiagrammes in Richtung der Pfeile. So ist die Folge „REPEAT anweisung; anweisung UNTIL ausdruck" korrekt. Als Programmstück wird sie in einer ebenfalls erlaubten anderen Anordnung

```
REPEAT
anweisung;
anweisung;
UNTIL ausdruck
```

sofort deutlich sichtbar.
Das Syntaxdiagramm für die Repeat-Anweisung enthält die klein geschriebenen deutschen Wörter „anweisung" und „ausdruck", die groß geschriebenen englischen Wörter REPEAT und UNTIL sowie das Semikolon. Während die Wörter REPEAT, UNTIL und das Semikolon unmittelbar im fertigen PASCAL-Programm erscheinen dürfen, verweisen „anweisung" und „ausdruck" auf weitere Syntaxdiagramme.
Die Namensgebung für Syntaxdiagramme ist beliebig. Günstig ist es aber, solche Namen zu wählen, die Assoziationen zur Sematik des entsprechenden Syntaxdiagramms herstellen. So wird bei der Repeat-Anweisung deutlich, daß in ihrem Inneren weitere Anweisungen eingelagert sein können.
In diesem Kurs wird durchgängig eine Darstellung gewählt, in der Syntaxdiagramme mit klein geschriebenen Wörtern der deutschen Sprache bezeichnet werden.

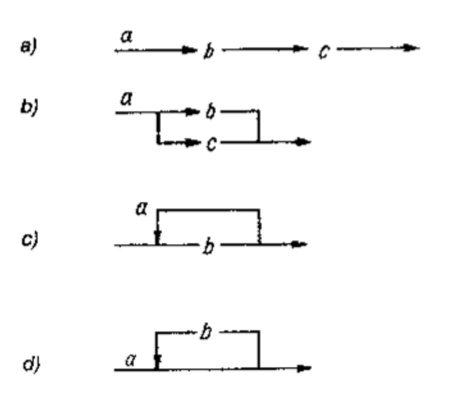


Bild 1.1 Grundformen von Syntaxdiagrammen
a) Sequenz
b) Alternative
c) Liste mit mindestens einem Element
d) Liste, die auch leer sein kann

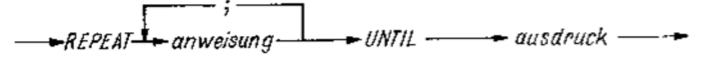
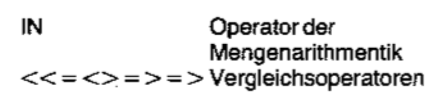


Bild 1.2 Syntaxdiagramm der Repeat-Anweisung

2. Grundelemente von PASCAL

2.1. Überblick

Die Grundelemente der Programmiersprache PASCAL können in die Klassen



- **Bezeichner**
- **Zahlen**
- **Zeichenketten**
- **Operatoren und spezielle Symbole**

eingeteilt werden.

2.2. Bezeichner

Bezeichner dienen zur Benennung von Objekten eines PASCAL-Programms. Die zu benennenden Objekte können Datentypen, Daten und Programmkodes sein.

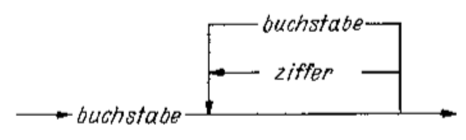


Bild 2.1 Syntaxdiagramm „bezeichner"

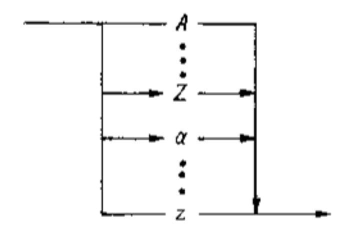


Bild 2.2 Syntaxdiagramm „buchstabe"



Bild 2.3 Syntaxdiagramm „ziffer"

Die Bildung von Bezeichnern zeigt das Syntaxdiagramm im Bild 2.1. Zur Substitution von „buchstabe" und „ziffer" sind die Syntaxdiagramme nach Bild 2.2. und Bild 2.3. heranzuziehen. Korrekte Bezeichner sind:

Max, x1, a23x, AnfWert, Count

Die Anzahl der signifikanten Stellen eines Bezeichners ist i. allg. begrenzt. Bei TURBO-PASCAL darf ein Bezeichner nicht länger als die Zeilenlänge von 127 Zeichen sein. Weiterhin werden innerhalb eines Bezeichners Groß- und Kleinbuchstaben nicht unterschieden. So sind z. B. die Bezeichner XY und xy gleich.
In PASCAL haben eine Reihe von Bezeichnern eine schon feststehende Bedeutung. Sie sind reserviert und dürfen vom Anwender zur Benennung seiner Objekte nicht benutzt werden:

ABSOLUTE(*)	FUNCTION	RECORD
AND	GOTO	REPEAT
ARRAY	IF	SET
BEGIN	IN	SHL(*)
CASE	INLINE(*)	SHR(*)
CONST	LABEL	STRING(*)
DIV	MOD	THEN
DO	NIL	TO
DOWNTO	NOT	TYPE
ELSE	OF	UNTIL
END	OR	VAR
EXTERNAL(*)	OVERLAY(*)	WHILE
FILE	PACKED	WITH
FOR	PROCEDURE	XOR(*)
FORWARD	PROGRAM	

Die durch (*) gekennzeichneten Bezeichner sind zusätzlich in TURBO-PASCAL reserviert.

2.3. Zahlen

Zahlen sind namenlose Datenobjekte mit einem festen numerischen Wert. In PASCAL werden ganze und reelle Zahlen unterschieden. Die Bildung erfolgt nach den Syntaxdiagrammen der Bilder 2.4 und 2.5. Korrekte Beispiele für ganze bzw. reelle Zahlen sind:

12, 10, 1000, 123
bzw.
1.0, 1.5E–03, 121E+14

Zur Unterstützung der maschinennahen Programmierung können ganze Zahlen in hexadezimaler Schreibweise angegeben werden. Es sind dann zusätzlich die Ziffern A, B . . ., F bzw. a, b, . . ., f erlaubt. Gekennzeichnet wird die hexadezimale Schreibweise durch ein vorangestelltes Dollarzeichen (bzw. das Zeichen ○)
○ 100, ○ 1FF, ○ FFFF

2.4. Zeichenketten

Eine Zeichenkette repräsentiert ein Datenobjekt, dessen Wert eine Folge von Zeichen des ASCII-Kodes ist. Innerhalb eines PASCAL-Programms kann sie ähnlich wie eine Zahl verwendet werden.
Die Bildung von korrekten Zeichenketten zeigt das Syntaxdiagramm im Bild 2.6. Dabei darf eine Zeichenkette nicht über das Zeilenende hinweg fortgesetzt werden. Korrekte Zeichenketten sind:

'Testprogramm Version 1.0', 'x:='

Als Spezialfall ist auch die leere Zeichenkette '' zugelassen

2.5. Operatoren und spezielle Symbole

Operatoren geben die mit den Datenobjekten durchzuführenden Verknüpfungen an. In PASCAL sind folgende Operatoren definiert:

:=	Zuweisungsoperator
+ – * /	Arithmetische Grundoperationen
DIV	Division ohne Rest
MOD	Divisionsrest
NOT AND OR	Grundoperationen der Booleschen Arithmetik
IN	Operator der Mengenarithmentik
<< = <> = > = >	Vergleichsoperatoren

Die Grundoperatoren verknüpfen in der Regel zwei Datenobjekte gleichen Typs und liefern wieder ein Resultat von diesem Typ. Vergleichsoperatoren ergeben unabhängig vom Typ der Operanden stets einen Booleschen Wert.
In TURBO-PASCAL gibt es eine Reihe von weiteren Operatoren. Sie werden später zusammen mit den Datentypen, auf die sie anwendbar sind, besprochen.
Weiterhin gibt es in PASCAL eine Reihe von speziellen Symbolen, die aus syntaktischen Gründen notwendig sind:

()	Klammerung der Parameterliste von Prozeduren bzw. Funktionen
[]	Klammerung der Indizes von Arrays
{} oder (**)	Klammerung von Kommentaren

Kommentare haben normalerweise keine Auswirkung auf ein Programm. In PASCAL ist es jedoch möglich, mit sogenannten Pseudokommentaren Compileroptions anzugeben, die die Arbeitsweise des Compilers beeinflussen. Eine Option veranlaßt den Compiler z. B., den Zugriff auf Arrays so zu übersetzen, daß die Überschreitung der Indexgrenzen zum definierten Programmabbruch führt. Es gibt weitere Compileroptions, die im Zusammenhang mit den entsprechenden Sprachelementen behandelt werden.
Compileroptions beginnen mit einem Dollarzeichen. Dann folgen ein Buchstabe, der die entsprechende Option charakterisiert und entweder ein Plus- oder ein Minuszeichen, das anzeigt, ob sie aktiviert werden soll oder nicht.

2.6. Format von PASCAL-Programmen

An das Format von PASCAL-Programmen werden nur zwei Forderungen gestellt:

– Die Grundelemente der Sprache sind durch mindestens ein Leerzeichen oder einen Kommentar zu trennen. Falls die Eindeutigkeit erhalten bleibt, kann das Leerzeichen auch entfallen.
– Bezeichner dürfen nicht durch Zeilenende zerteilt werden.

Insbesondere zur Verbesserung der Lesbarkeit können zwischen den Grundelementen beliebig viele Leerzeichen oder Leerzeilen eingefügt werden. Die PASCAL-Passage

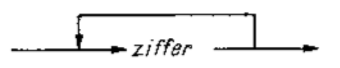


Bild 2.4 Syntaxdiagramm „ganze zahl"

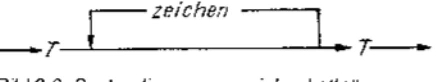


Bild 2.6 Syntaxdiagramm „zeichenkette"

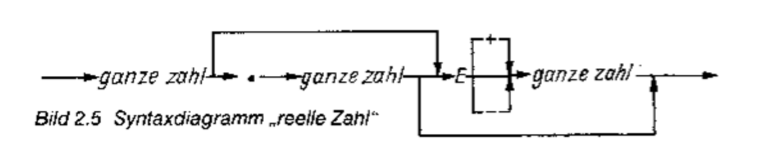


Bild 2.5 Syntaxdiagramm „reelle Zahl"

a + 1.0 bleibt beim Weglassen der Leerzeichen eindeutig. Das ist jedoch bei REPEAT A; B UNTIL C nicht so.

2.7. Programmbeispiel

Die Anordnung der einzelnen Grundelemente zu einem kompletten PASCAL-Programm zeigt der nachfolgende Programmtext. Das Programm tabelliert die Funktion
$y = f(x) = x^2$
für ganzzahlige x im Intervall 0 . . max.

```
PROGRAM Tabelle
                    (INPUT, OUTPUT);
VAR x, y, max.: INTEGER;
BEGIN
  READLN (INPUT, max);
  FOR x: = 0 TO max DO BEGIN
    y: = x*x;
    WRITELN (OUTPUT,'x: = ', x,
    'y: = ', y)
  END
END
```

Die Aufteilung des Programmtextes auf die einzelnen Zeilen kann auch in anderer Weise erfolgen. Ebenso ist das Einrücken von Zeilen nicht zwingend. Die gezeigte Form wurde ausschließlich mit dem Ziel gewählt, die inhaltliche Zusammengehörigkeit der Programmteile auch optisch zum Ausdruck zu bringen.
Zur Konstruktion des Programms wurden folgende Grundelemente verwendet:

Bezeichner:
PROGRAMM INPUT Tabelle VAR × y max INTEGER BEGIN READLN FOR TO DO WRITELN END
Zahlen:
0
Zeichenketten:
'x: = ''y: = '
Operatoren und spezielle Symbole:
; , : () : = *.

3. Datentypen

3.1. Überblick

Die Aufgabe von Datentypen ist es, Wertebereiche für Datenobjekte anzugeben. PASCAL stellt eine Reihe von *Standarddatentypen* zur Verfügung. Zusätzlich können vom Programmierer in gewissen Grenzen *problemangepaßte Datentypen* deklariert werden.

Die Standardtypen sind
- ganze Zahlen (INTEGER)
- reelle Zahlen (REAL)
- Boolesche Werte (BOOLEAN)
- Zeichenkodes des Rechners (CHAR)

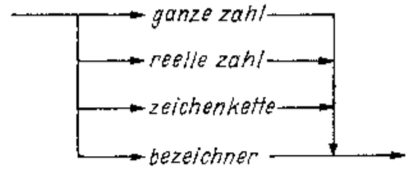


Bild 2.7 Syntaxdiagramm „konstante"

Die Wertebereiche der Standardtypen werden durch die Datenformate der zugrundeliegenden Gerätetechnik bestimmt. Hier gibt es Unterschiede zwischen den PASCAL-Systemen verschiedener Rechner.
Der Programmierer kann Aufzählungs- und Teilbereichstypen deklarieren. Für *Aufzählungstypen* sind die Elemente des Wertebereiches explizit zu nennen. Bei *Teilbereichstypen* wird eine Eingrenzung auf einen Teil eines umfassenderen Wertebereiches vorgenommen.
Als weiterer Datentyp werden sogenannte *Mengen* bereitgestellt.
Mit allen bisher genannten Datentypen lassen sich die Strukturen *File, Array* und *Rekord* bilden. Während Files und Arrays aus Datenobjekten gleichen Typs bestehen, können zu Rekords Datenobjekte unterschiedlichen Typs zusammengefaßt werden. Mit dem Datentyp File und darauf anwendbaren Prozeduren und Funktionen wird in PASCAL das gesamte Problem der Ein- und Ausgabe gelöst.
Pointertypen gestatten in Abhängigkeit vom Programmablauf das dynamische Anlegen von Datenobjekten aller bisher genannten Typen. Der Wert eines Pointertyps selbst ist eine Hauptspeicheradresse.
Die bisher genannten Datentypen gehören zum Standard-PASCAL. Darüber hinaus gibt es in einzelnen Systemen nützliche Erweiterungen. In TURBO-PASCAL sind das
- ganze Zahlen im Byteformat (BYTE)
- Zeichenketten (STRING).

3.2. Einordnung in die Sprache

Datentypen werden durch das Syntaxdiagramm „typ" deklariert. Es wird im Bild 3.1 gezeigt. Die Standard-, Aufzählungs- und Teilbereichstypen sind zum Syntaxdiagramm „einfacher typ" zusammengefaßt worden und werden im Bild 3.2 gezeigt. Das hat ausschließlich einen praktischen Grund: An verschiedenen Stellen der Sprache PASCAL sind nicht alle Typen von Datenobjekten zugelassen, sondern nur Standard-, Aufzählungs- und Teilbereichstypen.

4. Einfache Typen

4.1. Standardtypen

4.1.1. Datentypen INTEGER und REAL

Der Wertebereich des Datentyps INTEGER umfaßt positive und negative ganze Zahlen. Der Bereich selbst hängt von der internen Verarbeitungsbreite des Rechners ab. Bei den Büro- und Arbeitsplatzrechnern ist sie 16 Bit. Damit wird ein Bereich –32 768 . . . . 32 767 überstrichen.
Die interne Darstellung des Standardtyps REAL erfolgt in Mantisse und Exponent. Die verwendete Anzahl von Bytes ist bei den einzelnen PASCAL-Systemen unterschiedlich. Bei TURBO-PASCAL sind es sechs: eins für den Exponenten und fünf für die Mantisse. Da die Mantisse selbst normalisiert abgespeichert wird, entspricht das einer Darstellung in 40 Bit. Dem entsprechen elf signifikante Stellen in der Dezimalschreibweise.
Für alle Datentypen ist die Ausführung der arithmetischen Grundoperation +, –, * und /

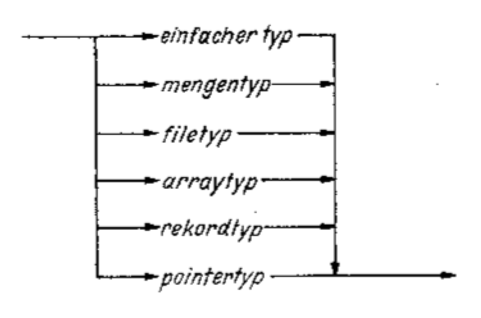


Bild 3.1 Syntaxdiagramm „typ"

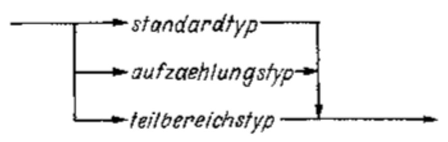


Bild 3.2 Syntaxdiagramm „einfacher typ"

und der Vergleiche <, <=, <>, =, > = und > möglich. Für den Datentyp INTEGER gibt es zusätzlich die ganzzahlige Division DIV und den Divisionsrest MOD.
Bei Büro- und Arbeitsplatzrechnern werden die arithmetischen Grundoperationen für den Datentyp REAL softwaremäßig realisiert. Sie sind deshalb deutlich langsamer.
Das Verhalten von PASCAL-Programmen bei *Überschreitung des erlaubten Wertebereiches* ist vom Datentyp abhängig. Bei INTEGER äußert sich eine Überschreitung wegen der internen Darstellung negativer Zahlen im Zweierkomplement als Vorzeichenwechsel und wird bei der Programmabarbeitung nicht „entdeckt".
Bei *Unterschreitung* der betragsmäßig kleinsten REAL-Zahl wird das entsprechende Datenelement auf den Wert 0.0 gesetzt und die Abarbeitung des Programms fortgesetzt. Überschreitet der Betrag den größten darstellbaren Wert, bricht das Programm mit einer entsprechenden Fehlerausschrift ab.
In TURBO-PASCAL werden bestimmte Operationen des Maschinenbefehlssystems auf das Niveau von PASCAL gehoben. Sie stehen als Operatoren SHL, SHR, OR und XOR zum Schieben bzw. für Bitoperationen zur Verfügung und können auf Datenelemente vom Typ INTEGER angewendet werden.

4.2.1. Datentyp BOOLEAN

Den Wertebereich des Datentyps BOOLEAN bilden die Wahrheitswerte FALSE und TRUE. Es sind die Operationen NOT, OR und AND sowie alle Vergleiche möglich, wobei FALSE < TRUE ist.
Intern wird ein Byte zur Darstellung benötigt, wobei 00 den Wert FALSE und 01 TRUE repräsentiert.

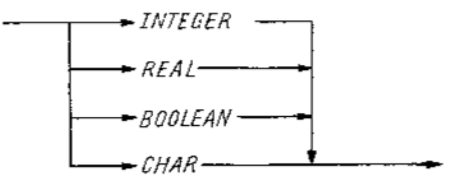


Bild 4.1 Syntaxdiagramm „standardtyp"

4.1.3. Datentyp CHAR

Der Wertebereich des Datentyps CHAR ist der Zeichenkode des jeweiligen Rechners. Beim PC 1715 ist das der ASCII-Kode. Die interne Darstellung erfordert ein Byte. Mit Datenobjekten zum Typ CHAR können keine arithmetischen Verknüpfungen durchgeführt werden. Zuweisung und Vergleichsoperationen sind erlaubt. Vergleichsoperationen orientieren sich an der internen Kodierung der einzelnen Zeichen.
Konstanten vom Typ CHAR werden in Apostrophe eingeschlossen, z. B. 'A', 'B', '+', '*', '>', '0', '1' usw.

4.2. Aufzählungstypen

Aufzählungstypen werden durch explizite Angabe ihres Wertebereiches deklariert. Bild 4.2 zeigt die Syntax. Die durch Komma getrennten Bezeichner bilden den Wertebereich. Ein Beispiel zeigt dies:

(ROT, BLAU, GRUEN).

Ein Datenobjekt von diesem Aufzählungstyp kann nur einen der drei möglichen Werte ROT oder BLAU oder GRUEN annehmen.

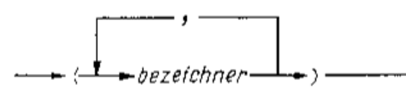


Bild 4.2 Syntaxdiagramm „aufzaehlungstyp"

Weitere Beispiele für Aufzählungstypen sind

(Mo, Di, Mi, Don, Fr, Sa, So)
(kleiner, gleich, grösser)
(Lesen, Schreiben, Positionieren).

Durch die Reihenfolge bei der Deklaration wird gleichzeitig eine Ordnung zwischen den einzelnen Werten eines Aufzählungstyps eingeführt. Für die genannten Beispiele gilt:

ROT < BLAU < GRUEN
Mo < Di < Mi < Don < Fr < Sa < So
usw.

Die Ausführung arithmetischer Operationen ist mit Aufzählungstypen nicht möglich. Es sind nur Zuweisung und Vergleichsoperationen erlaubt.
Werden mehrere Aufzählungstypen deklariert, müssen die Wertebereiche disjunkt sein. Zum Beispiel ist

(ROT, BLAU, GRUEN)
(GELB, ORANGE)
erlaubt, aber
(ROT, BLAU, GRUEN)
(ROT, GELB)

nicht, da der Wert ROT nicht zwei Datentypen angehören darf.
Intern werden Aufzählungstypen in einem Byte dargestellt. Dadurch ist die maximale Anzahl von Werten auf 256 beschränkt.

4.3. Teilbereichstypen

Teilbereichstypen ermöglichen die Eingrenzung des Wertebereiches von Datenobjekten auf einen Teil eines sogenannten Basistyps. Basistypen können sein INTEGER, BOOLEAN, CHAR und Aufzählungstypen. Die Deklaration zeigt das Syntaxdiagramm im Bild 4.3. Die Konstanten müssen von einem der erlaubten Basistypen sein.

——► *konstante* ——► .. ——► *konstante* ——►

Bild 4.3 Syntaxdiagramm „teilbereichstyp"

Es folgen einige Beispiele:

Teilbereiche des Typs INTEGER
1...12
−3...3
0...100
Teilbereiche des Typs CHAR
'A'...'Z'
'0'...'9'
Teilbereiche des Aufzählungstyps
(Mo, Di, Mi, Don, Fr, Sa, So)
Mo...Fr
Sa...So
Di...Do

Intern werden Teilbereichstypen wie ihr Basistyp dargestellt.
PASCAL-Systeme sichern die Einhaltung des Wertebereiches von Teilbereichstypen. Soll einem Datenobjekt ein nicht erlaubter Wert zugewiesen werden, wird die Abarbeitung des Programms mit einer entsprechenden Fehlermeldung beendet. Diese Überprüfung erhöht den Speicherplatzbedarf und die Laufzeit eines Programms. Deshalb kann sie durch die Compileroption OR- bzw. OR+ aus- bzw. eingeschaltet werden.

5. Struktur eines PASCAL-Programms

5.1. Überblick

Der grundlegende Baustein eines PASCAL-Programms ist ein sogenannter *Block*. Seine Syntax zeigt Bild 5.1. Ein Block besteht aus Deklarations- und Anweisungsteil. Mit ihm können Prozeduren und Funktionen sowie das Hauptprogramm selbst gebildet werden.

——► *deklarationsteil* ——► *anweisungsteil* ——►

Bild 5.1 Syntaxdiagramm „block"

Jedes PASCAL-Programm einschließlich aller benötigten Prozeduren und Funktionen ist eine einzige *Übersetzungseinheit*. Das ermöglicht dem Compiler eine vollständige Kontrolle der Konsistenz aller Programmteile.

5.2. Deklarationsteil
5.2.1. Überblick

Das Bild 5.2. zeigt die Syntax des Deklarationsteils. Je nachdem, ob im Syntaxdiagramm die entsprechende waagerechte Alternative gewählt wird oder nicht, kann er Marken-, Konstanten-, Typen-, Variablen- sowie Prozedur- und Funktionsdeklarationen enthalten.
Während Marken-, Konstanten-, Typen- und Variablendeklarationen höchstens einmal und dann in der genannten Reihenfolge auftreten, können Prozedur- und Funktionsdeklarationen mehrmals vorhanden sein.

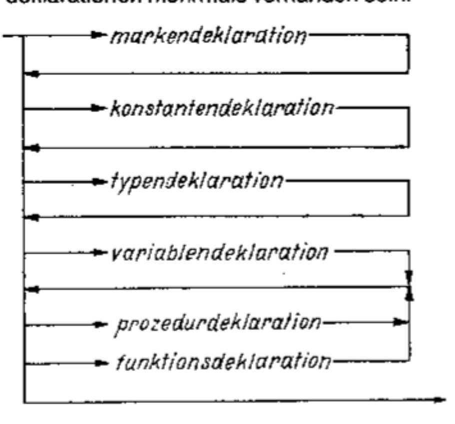


Bild 5.2 Syntaxdiagramm „deklarationsteil"

Das Syntaxdiagramm „deklarationsteil" läßt sich auch traversieren, ohne etwas zu deklarieren. Dann enthält der entsprechende Block keine Deklarationen.
In PASCAL herrscht *Deklarationszwang*. Alle Bezeichner müssen vor ihrer ersten Verwendung deklariert werden. Es gibt nur für den Datentyp Pointer eine Ausnahme. Bei TURBO-PASCAL ist die Syntax des Deklarationsteils geringfügig anders. Marken-, Konstanten-, Typen- und Variablendeklarationen dürfen mehrfach und auch gemischt auftreten. Bezeichner müssen aber auch dort vor ihrer Verwendung deklariert sein.
In den nachfolgenden Unterpunkten werden Marken-, Konstanten-, Typen- und Variablendeklarationen besprochen. Prozedur- und Funktionsdeklarationen werden solange aufgeschoben, bis der Anweisungsteil behandelt ist.

5.2.2. Markendeklaration

In PASCAL können beliebige Anweisungen mit einer Marke versehen werden. Mit der Goto-Anweisung kann dann unbedingt dorthin verzweigt werden.

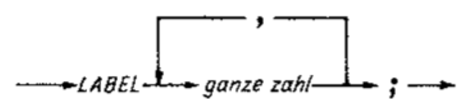


Bild 5.3 Syntaxdiagramm „markendeklaration"

Marken sind in PASCAL ganze Zahlen. Das Syntaxdiagramm in Bild 5.3 zeigt die Markendeklaration. Sie wird durch den reservierten Bezeichner LABEL eingeleitet. Beispiele sind

LABEL 99;
LABEL 10, 20, 30;

In TURBO-PASCAL können auch Bezeichner als Marken deklariert werden.

(Fortsetzung folgt)

Automatische Erzeugung von Syntaxanalysatoren

Dr. Uwe Hübner
VEB Zentrum für Forschung und Technologie Nachrichtenelektronik, Berlin

1. Zielstellung

Die Notwendigkeit einer Effektivitätserhöhung beim Software-Herstellungsprozeß ist unbestritten. Eine wesentliche Methode ist dabei die Verringerung des Niveauunterschiedes zwischen der Problembeschreibung und der Implementierung in einer gewählten Quellsprache. Die Verwendung geeigneter höherer Programmiersprachen stellte bereits einen wesentlichen Effektivitätsgewinn für die Softwareherstellung dar. Man kann die Annäherung an die Problembeschreibung weiter verbessern, indem man von der ablauforientierten (prozeduralen) Darstellung zu einer deklarativen Darstellung übergeht. Einen recht hohen Stand hat diese Technik bei der Programmierung der Syntaxanalyse eines Eingabedatenstroms erreicht.
Eine wichtige Anwendung ist die Herstellung von Übersetzerprogrammen; aus diesem Grund bilden Werkzeuge dieser Art auch oft den Kern sogenannter *Compilergeneratoren* oder *Compiler-Compiler*, da tatsächlich eine (wenigstens teilweise) Automatisierung der Compilerherstellung eintritt (/1/ bis /4/).
Das hier vorgestellte Werkzeug SYNA gehört ebenfalls zu dieser Gruppe von Hilfsmitteln, wobei die Belange „kleiner" Rechentechnik besonders berücksichtigt werden, das heißt, das Werkzeug ist für praktisch interessierende Grammatikumfänge bereits auf 8-Bit-Bürocomputern (A 5120 mit 64 KByte Hauptspeicher und Betriebssystem UDOS) nutzbar. Die mit SYNA erzeugten Syntaxanalysatoren (bzw. Compiler) weisen bezüglich Speicherplatzbedarf und Laufzeit keine wesentlichen Abweichungen gegenüber „handgefertigten" Lösungen auf.
Die Herstellung von Compilern für allgemeine höhere Programmiersprachen ist nicht der einzige Einsatzfall von Syntaxanalyse-Generatoren. Weitere Anwendungsgebiete sind die Auswertung komplexer Kommandosprachen, syntaxgesteuerte Druckformatierer oder Editoren, Analysen von Text bzw. Programmdokumentationen unter verschiedenen Gesichtspunkten (/5/ bis /7/).
SYNA steht auch für das Betriebssystem MUTOS 8000 des Bürocomputers A 5120.16 zur Verfügung, dort gibt es allerdings bereits leistungsfähigere Werkzeuge gleicher Zielstellung /1/.

2. Arbeitsweise

Da ein Übersetzer nicht nur aus der Syntaxanalyse besteht, wurde SYNA in eine prozedurale höhere Programmiersprache eingebettet, das heißt, der Quelltext des Übersetzerprogramms besteht aus der Eingabesyntax-Beschreibung durch eine Grammatik und Programmteilen in einer höheren Programmiersprache. An dieser Stelle eignen sich Systemprogrammiersprachen besonders gut, weil damit Aufgaben wie Bezeichnerlistenorganisation o. ä. gut ausgedrückt werden können. Gewählt wurde die Sprache PLZ/SYS /8/, /9/, weil auch auf 8-Bit-Rechentechnik leistungsfähige und zuverlässige Implementierungen vorliegen (die Sprache C ist an dieser Stelle ebenfalls gut geeignet).
SYNA arbeitet als „Filter" und setzt die Grammatik-Beschreibung in ein gewöhnliches PLZ/SYS-Programm um; es werden Datenstrukturen erzeugt, die einer vorgegebenen Syntaxanalyse-Prozedur PARSE als Arbeitsvorschrift dienen. Die PLZ/SYS-Programmteile werden ohne Änderung in die Ausgabe übernommen (Bild 1). Diese wird dann vom vorhandenen PLZ/SYS-Übersetzersystem in ein ablauffähiges Programm umgesetzt.
Die Syntaxanalyse beruht auf dem Prinzip des rekursiven Abstiegs, wobei allerdings hier nicht jeder Regel eine eigene Prozedur zugeordnet, sondern eine Datenstruktur interpretativ bearbeitet wird. Auf dieses Funktionsprinzip soll hier nicht näher eingegangen werden, es ist z. B. in /10/ detailliert beschrieben.
Die vorgegebene Grammatik muß vom Typ LL(1) sein, daß heißt, bei alternativen Regelteilen muß vom ersten Element der Alternative entschieden werden, welcher Zweig der Alternative vorliegt. Trotz dieser Einschränkung können mit LL(1)-Grammatiken viele praktisch relevante Sprachen (insbesondere solche vom Pascal-Typ) beschrieben werden. Für die Umformung allgemeiner Grammatiken in LL(1)-Grammatiken existiert keine allgemeine Vorschrift, eine solche läßt sich aber für bestimmte Detailprobleme angeben /11/; in vielen Fällen ist die LL(1)-Eigenschaft auch durch eine geeignete Funktionsteilung mit Lexikanalyse bzw. Semantikprozeduren herstellbar (beispielsweise, indem bereits der Lexikanalysator zwischen undefinierten und bereits definierten Namen unterscheidet).

3. Grammatik-Beschreibungssprache

Die Eingabesprache von SYNA enthält als Hauptelemente:

1. Eine Liste der lexikalischen Einheiten der Sprache (d. h. die Resultatmenge der lexikalischen Analyse)
2. Eine Liste von Regeln für den syntaktischen Aufbau und deren Umsetzung.

Ferner können Programmteile in der höheren Systemprogrammiersprache PLZ/SYS angegeben werden. Die neuen Sprachelemente werden in die Basissprache PLZ/SYS als neue Deklarationsklassen eingefügt; neben den bisherigen Klassen, TYPE, CONSTANT, INTERNAL, GLOBAL, EXTERNAL, werden die neuen Klassen TERMINALS und RULES eingeführt. Diese beiden Klassen dürfen in einem Modul je einmal und nur in der angegebenen Reihenfolge auftreten.
Die nachfolgenden Erläuterungen beziehen sich auf die Syntaxgrafen in Bild 2. Dabei werden die verschiedenen semantischen Interpretationen von "..._name" sowie PLZ/SYS-Programmteile ("semantic_clause") als Elementareinheiten betrachtet (runde Umrahmung).

3.1. Lexikalische Einheiten – TERMINALS

Nach TERMINALS sind alle lexikalischen Einheiten der zu analysierenden Sprache zu benennen, mit Ausnahme von einzelnen Textzeichen, die in RULES direkt angegeben werden (z. B. '+').

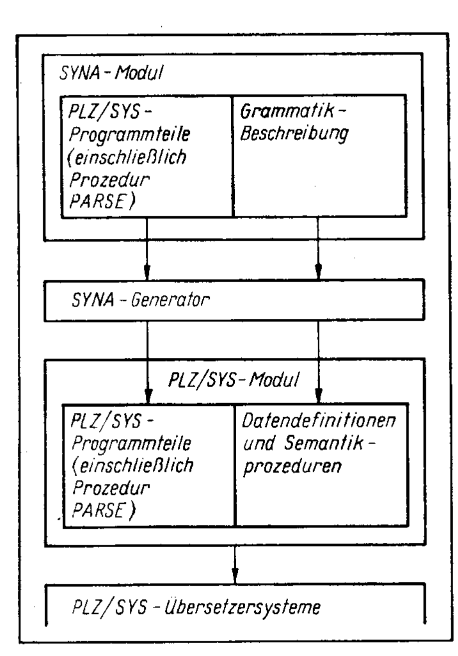


Bild 1 Arbeitsweise von SYNA

```
terminals_declaration:
      TERMINALS        lexical_unit_name

rule_declaration:
      RULES            rule
rule:
      rule_name    :    rule_description
rule_description:
      rule_alternative
            |
rule_alternative:
      rule_part
      optional_rule_part
      repeated_rule_part
optional_rule_part:
      [     rule_part     ]
repeated_rule_part:
      (     rule_part     )
            |
rule_part:
      rule_name
      lexical_unit_name
      '   character   '
      {   semantic_clause   }
          semantic_clause_name
```

Bild 2 Syntax der Grammatik-Beschreibung

```
terminals    CALL DCL ELSE END FI IF INT MODULE PROC THEN
             EQ NAME NUMBER EOF.
rules
SYNTAX : NAME ':' MODULE data_declaration
       (procedure_declaration) END NAME ';' EOF.

data_declaration : DCL data_decl1 (',' data_decl1) ';'.

data_decl1 : NAME                          {out(#'a DEFS 2 %r')}
             INT .

procedure_declaration : NAME                        {out(#'a:%r')}
                        ':' PROC '(' ')' ';'
                        (action) END NAME ';' {out(#' RET%r')}.

action :        ifaction
              | callaction
              | assignaction.

ifaction : IF                          {MARVAL += 1 SYNVAL:=MARVAL}
              condition THEN           {out(#' JP ELs %r')}
              (action)
           ELSE            {out(#' JP FIs %r') out (#'ELs: %r')}
              (action)
           FI                          {out(#'FIs: %r')}
           ';'.

callaction : CALL NAME                           {out(#' CALL a %r')}
             '(' ')' ';'.

assignaction : NAME                                      {move_a_to_b}
               EQ expression                   {out(#' LD (b),HL%r')}
               ';'.

expression : operand
             [operator {SYNVAL := LASTSYM out(#' EX DE,HL %r')}
             operand
                {if SYNVAL case '+' then out(#' ADD HL,DE %r')
                           case '-' then out(#' AND A %r')
                                         out(#' EX DE,HL %r')
                                         out(#' SBC HL,DE %r')
                 fi}].

operator : '+' | '-'.

condition : operand
            condoperator {SYNVAL := LASTSYM out(#' EX DE,HL %r')}
            operand {out(#' AND A %r')
                     out(#' SBC HL,DE %r')
                     if SYNVAL case '=' then out(#' JR Z,¤+5%r')
                                case '<' then out(#' JR NC,¤+5%r')
                     fi}.

condoperator : '=' | '<'.

operand :          NAME                    {out(#' LD HL,(a)%r')}
          |        NUMBER                    {out(#' LD HL,a%r')}..
```

Bild 3
Anwendungsbeispiel

Dr. Uwe Hübner (30) studierte an der Technischen Universität Karl-Marx-Stadt Informationstechnik. Nach einem Forschungsstudium promovierte er mit einer Arbeit zu dienstintegrierten Kommunikationsnetzen. Er arbeitet gegenwärtig als Abteilungsleiter für Softwareentwicklung im VEB Zentrum für Forschung und Technologie Nachrichtenelektronik.

Die Festlegung, was als lexikalische Einheit angesehen wird, stellt einen Kompromiß bezüglich der Funktionsteilung zwischen Lexik- und Syntaxanalyse dar. Wenn beispielsweise jedes einzelne Textzeichen als lexikalische Einheit betrachtet wird, erhält man zwar einen sehr einfachen Lexikanalysator, dafür aber einen erheblichen Aufwand in der Syntaxanalyse. Im allgemeinen hat es sich bewährt, Schlüsselwörter, Bezeichner und Sonderzeichen als lexikalische Einheiten zu betrachten.
Unter TERMINALS sind also die Schlüsselwörter einer Sprache aufzuzählen (z. B. IF, THEN), ferner alle Sprachsymbole, die aus mehreren Sonderzeichen bestehen (z. B. ,:=') und alle abstrakten Klassen von lexikalischen Einheiten (z. B. NAME, NUMBER, STRING).

Beispiel

```
terminals
!Schlüsselworte!        IF THEN ELSE FI
!Sondersymbole für
!:=, +=, -=!            EQ PE ME
!abstrakte Klassen!     NAME NUMBER EOF.
```

3.2. Syntaktische Regeln – RULES

Die Regeln der zu verarbeitenden Grammatik werden in einer modifizierten Backus-Naur-Form angegeben. Die Regel der obersten Ebene muß als erste angegeben werden; ansonsten ist die Reihenfolge beliebig, auch Rekursion ist im Rahmen der Möglichkeiten einer LL(1)-Grammatik zulässig. Innerhalb einer Regelbeschreibung ("rule_description") dürfen als Elemente Regelnamen, Namen lexikalischer Einheiten und einzelner Textzeichen (in Apostrophe eingeschlossen) auftreten.

Beispiel:

```
rules
programm:   NAME MODULE declarations
            END NAME';'..
```

In einer Regelbeschreibung kann nun nicht nur eine starre Folge lexikalischer Einheiten bzw. grammatischer Regeln angegeben werden, sondern es ist zulässig, mehrere Alternativen für die Bildung der Regel anzugeben:

Beispiel:

```
action:     IF ifaction
          | DO doaction
          | caseaction
          | assignaction.
```

Zur Erfüllung der LL(1)-Bedingungen müssen sich dabei die ersten lexikalischen Einheiten jeder "rule_alternative" unterscheiden. Nicht zulässig ist beispielsweise:

```
expression:    operand '+' operand
             | operand '-' operand.
```

Dieser Fall ist nach Auflösung in zwei Regeln, aber ebenfalls mit einer LL(1)-Grammatik, darstellbar:

```
expression    :  operand expression_1.
expression_1 :   '+' operand
               | '-' operand.
```

Innerhalb jeder "rule_alternative" ist nun noch spezifizierbar, daß Folgen von Elementen 0 ... 1mal vorhanden sein dürfen (durch eckige Klammern) oder 0 ... beliebig oft vorhanden sein dürfen (durch runde Klammern). Bei der Wiederholung ("repeated_rule_part") ist es weiterhin möglich, mehrere Alternativen für "rule_part" zu spezifizieren. Die LL(1)-Eigenschaft erfordert wiederum, daß mit dem ersten Element eines wahlweise bzw. beliebig oft vorhandenen Regelteils entschieden werden kann, ob der wahlweise Teil bzw. eine weitere Wiederholung vorhanden ist.
Als einfacher Regelteil können eine Folge von Namen lexikalischer Einheiten oder anderer Regeln, einzelne Textzeichen oder in geschweifte Klammern gesetzte Semantikanweisungen auftreten.
Die verwendeten Namen lexikalischer Einheiten ("lexical_unit_name") müssen vorher unter TERMINALS definiert worden sein; verwendete Regeln ("rule_name") müssen definiert sein (nicht notwendigerweise vorher). Für einzelne Textzeichen sind alle Möglichkeiten von PLZ/SYS zugelassen, z. B. Sonderzeichen oder beliebige Hexadezimalkodierungen (z. B. '%r', '%7F').
Semantikanweisungen ("semantic_clause") enthalten Programmtext in PLZ/SYS. Diese Programmteile werden ausgeführt, sobald die entsprechende Stelle der Regel bei der Analyse erreicht wird, das heißt, Semantikanweisungen sind nicht nur am Regelende zulässig. Wenn Semantikanweisungen sehr oft in identischer Form auftreten, ist es zweckmäßig, dafür eine (parameterlose) Prozedur zu formulieren und deren Namen nach '*' als "semantic_clause_name" anzugeben. Dadurch wird der Umfang des von SYNA erzeugten Analyseprogramms verringert.

3. Anwendungsbeispiel

Zur Erläuterung der Möglichkeiten und der Handhabung von SYNA wird nachfolgend eine sehr einfache Anwendung beschrieben. Es soll eine stark eingeschränkte Untermenge der höheren Progammiersprache CHILL /12/ in Assemblerkode für den bekannten Mikroprozessor U880 übersetzt werden. Es sei darauf hingewiesen, daß praktisch nutzbare CHILL-Compiler wesentlich größere Sprachumfänge aufweisen müssen; dies ist mit SYNA bereits erfolgreich realisiert worden.
Ein Modul der Beispielsprache besteht aus Daten- und Prozedurdeklarationen. Es sind nur Variablen des Modus INT (ganze Zahlen) vorgesehen. Prozeduren haben keine Parameter, innerhalb von Prozeduren sind Alternativanweisungen, Prozeduraufrufe und Wertzuweisungen möglich. Die Beispielsyntax wurde in die in Bild 2 dargestellte SYNA-Grammatikvorgabe umgesetzt.

Der Inhalt der Semantikteile ist für SYNA nicht relevant, zum Verständnis des Beispiels sind einige Erläuterungen notwendig: Kern der Semantikteile ist hier die Prozedur "out", die eine Ausgabe des als Parameter angegebenen Textes (bis zum Zeilenendezeichen '%r') realisiert. Mit Kleinbuchstaben werden dabei zu substituierende Parameter bezeichnet:

- a – letztes Sybol des Quelltextes
- b – ein mit move_a_to_b gerettetes Symbol des Quelltextes
- s – der Wert von SYNVAL als Hexadezimalzahl, verwendet zur Erzeugung von Marken im Ausgabetext

LASTSYM bezeichnet das zuletzt akzeptierte SYMBOL, SYNVAL eine auf jeder Regelebene individuell verwendbare Variable und MARVAL einen Bezugswert für die Erzeugung von Marken im Ausgabetext.

Literatur

/1/ Johnson, S. C.: Language Development Tools on the Unix System. Computer 13 (1980) 8, 16–21

/2/ Schilling, A.: Das Compiler-Writing-System. edv-aspekte 4 (1985) 3, 24–28

/3/ Moessenboeck, H. Ein einfacher Compiler-Compiler für Mikrorechner. Elektronische Rechenanlagen 26 (1984) 4, 186–193

/4/ Aho, A. V.: Translator Writing Systems: Where do they stand?, Computer 13 (1980) 8; 9–14

/5/ Rubin, L. F.: Syntax-Directed Pretty Printing – A First Step Towards a Syntax-Directed Editor. IEEE-SE 9 (1983) 2, 119–127

/6/ Blaschek, G.: Statische Programmanalyse. Elektronische Rechenanlagen 27 (1985) 2, 89–95

/7/ Rechenburg, P.: Attributierte Grammatiken als Methode der Softwaretechnik. Elektronische Rechenanlagen 26 (1984) 3, 111–118

/8/ Conway, R., u. a.: Introduction to Microprocessor Programming Using PLZ. Winthrop Publishers, Cambridge, 1979

/9/ Anwenderdokumentation PLZ/SYS. VEB Robotron Buchungsmaschinenwerk, Karl-Marx-Stadt, 1984

/10/ Waite, W. M.; Goos, G.: Compiler Construction. Springer, New York, 1984

/11/ Aho, A. V.; Ullman, I. D.: Principles of Compiler Design. Addison-Wesley, Reading, 1978

/12/ CCITT High Level Language (CHILL). Recommendation Z.200, Geneva, 1985

☒ KONTAKT ☏

VEB Zentrum für Forschung und Technologie Nachrichtenelektronik,
Edisonstraße 63, Berlin,
1160; Tel. 6381 2241.

BASIC-Programm zur Stereo-Darstellung von Molekülen

Dr. Wolfgang Brandt,
Martin-Luther-Universität Halle-Wittenberg,
Sektion Chemie

Die grafische Darstellung von Molekülen ist für viele chemische und biochemische Probleme hilfreich.

Für Großrechner existiert schon eine Vielzahl geeigneter Programme (PLUTO, ORTEP usw.), welche das Zeichnen beliebiger Molekülstrukturen erlauben. Jedoch ist die Anwendung dieser Programme als Routinemethoden z. Z. mit relativ viel Rechen- und Zeitaufwand verbunden.

Es wurde deshalb ein BASIC-Programm für den PC 1715 mit EPSON-Drucker zur schnellen Veranschaulichung von Molekülen entwickelt. Da die Rechenzeit des compilierten und gelinkten Programms selbst für größere Moleküle (maximal 100 Atome) nur etwa sieben Minuten für eine Stereo-Zeichnung beträgt, können in kurzer Zeit viele Strukturen dargestellt werden. Gleichzeitig kann eine geeignete Ansicht von Molekülen für spätere PLUTO-Zeichnungen gesucht werden. Für das Programm werden die kartesischen Koordinaten der Atome sowie eine Verknüpfungsmatrix benötigt. Vorgelagerte und hier nicht aufgeführte Programme (zu erfragen beim Autor) erlauben die Berechnung der kartesischen Koordinaten aus internen Koordinaten (Bindungslängen, Bindungswinkeln und Torsionswinkeln) sowie die beliebige Rotation von Molekülen oder Molekülteilen. Damit ist die Modellierung insbesondere von Biomolekülen leicht möglich. Aufgrund des zur Verfügung stehenden Speicherplatzes darf die Zahl der zu zeichnenden Punkte jedoch nicht 4550 übersteigen, was jedoch bisher noch in keinem Beispiel erreicht wurde.

☒ KONTAKT ☏

Martin-Luther-Universität Halle, Sektion Chemie,
Weinbergweg 16, Halle, 4050;
Tel. 622213


```
10 CLEAR: PRINT CHR$(12)
20 LPRINT CHR$(27)CHR$(64);
30 INPUT"Name des Molekuels ";A$
40 INPUT"Weitere Kommentare ";G$
50 DIM BI(100,2): DIM C3(200): DIM C2(200)
60 DIM B(200,3): DIM Z(400): DIM C(4550): DIM CH(200)
70 PRINT"Eingabe per Hand oder Diskette H/D ?"
80 L$=INKEY$:IF L$=""THEN 80
90 IF L$="H" THEN 200
100 INPUT"Dateiname ";C$
110 PRINT"Laufwerk A/B  Diskette einlegen !!!"
120 Z$=INKEY$: IF Z$="" THEN 120
130 D$=Z$+":"+C$+".MOL":OPEN "I",#1,D$: INPUT#1,N
140 FOR I=1 TO N: FOR J=1 TO 3: INPUT#1,B(I,J): NEXT J,I
150 CLOSE #1
160 PRINT"Ausdrucken der kartesischen Koordinaten J/N ?"
170 L$=INKEY$: IF L$="" THEN 170
180 IF L$="J" THEN GOSUB 1060
190 GOTO 240
200 INPUT"Anzahl der Atome";N
210 FOR I=1 TO N: PRINT "ATOM NR.";I
220 INPUT B(I,1),B(I,2),B(I,3)
230 NEXT I
240 PRINT"Einlesen der Bindungsmatrix von Diskette J/N ?"
250 L$=INKEY$:IF L$="" THEN 250
260 IF L$="N" THEN 340
270 INPUT"Name der Datei ";E$
280 D$=Z$+":"+E$+".BIN":OPEN "I",#1,D$: INPUT#1,M
290 FOR I=1 TO M: FOR J=1 TO 2: INPUT#1,BI(I,J): NEXT J,I
300 CLOSE #1
310 PRINT"Ausdrucken der Bindungsmatrix oder Korrektur J/N ?"
320 L$=INKEY$: IF L$=""THEN 320
330 IF L$="J" THEN  1090: ELSE 470
340 INPUT"Anzahl der Bindungen ";M
350 PRINT"BINDUNGEN I,J"
360 FOR I=1 TO M: INPUT BI(I,1),BI(I,2): NEXT I
370 GOTO 310
380 PRINT"Abspeichern der Bindungsmatrix J/N ?"
390 L$=INKEY$: IF L$="" THEN 390
400 IF L$="N" THEN 470
410 INPUT"Name der neuen Datei ";C$
420 PRINT"Auf welches Laufwerk A/B ?"
430 D$=INKEY$: IF D$="" THEN 430
440 E$=D$+":"+C$+".BIN": OPEN "O",#1,E$: PRINT#1,M
450 FOR I=1 TO M: FOR J=1 TO 2: PRINT#1,BI(I,J)
460 NEXT J,I: CLOSE #1
470 INPUT"Darstellung der XY, XZ oder YZ Ebene ";D$
480 LPRINT CHR$(CR);
490 LPRINT CHR$(27)"K";
500 LPRINT A$:LPRINT:LPRINT CHR$(27)"F";
510 LPRINT G$
520 IF D$="XY"THEN P=1: T=2: Q=3:SI=1
530 IF D$="YZ"THEN P=3: T=2: Q=1:SI=-1
540 IF D$="XZ"THEN P=1: T=3: Q=2:SI=1
550 XK=0: XG=0: YK=0: YG=0:NN=N*2: GOSUB 1170
560 FOR I=1 TO NN
570 IF B(I,P)<XK THEN XK=B(I,P)
580 IF B(I,P)>XG THEN XG=B(I,P)
590 IF B(I,T)<YK THEN YK=B(I,T)
600 IF B(I,T)>YG THEN YG=B(I,T)
610 NEXT I
620 XD=XG-XK: YD=YG-YK
630 IF XD>=YD THEN F=XD: ELSE F=YD
640 F=190/F: XK=XK*F: YK=YK*F
650 FOR I=1 TO NN
660 B(I,P)=INT(B(I,P)*F-XK+5)
670 B(I,T)=INT(B(I,T)*F-YK+5)
680 B(I,Q)=INT(B(I,Q)*F)
690 NEXT I
700 FOR I=1 TO M: FOR V=0 TO 1
710 L=BI(I,1)+V*N: K=BI(I,2)+V*N
720 X=B(L,P)-B(K,P): Y=B(L,T)-B(K,T): R=SQR(X^2+Y^2)
730 IF R=0 THEN 840
740 X=B(L,P)-B(K,P): Y=B(L,T)-B(K,T)
750 IF ABS(X)>=ABS(Y) THEN H=ABS(X): ELSE H=ABS(Y)
760 X=X/H: Y=Y/H
770 FOR J=0 TO H
780 SX=INT(B(K,P)+J*X)+V*200
790 SY=INT(B(K,T)+J*Y)
800 IF HH=1 THEN 820
810 CH(SY)=CH(SY)+1: GOTO 830
820 C3(SY)=C3(SY)+1: K3=C2(SY-1)+C3(SY): C(K3)=SX
830 NEXT J,V
840 NEXT I
850 IF HH=0 THEN 1210
860 B$=A$+" in der "+D$+" Ebene ":LPRINT B$
870 LPRINT:LPRINT:LPRINT
880 FOR I=200 TO 1 STEP -1
890 IF CH(I)=0 THEN 960
900 K1=C2(I-1): K2=C2(I)
910 IF K2=0 THEN 960
920 FOR J=K1 TO K2: L=C(J): Z(L)=1: NEXT J
930 LPRINT CHR$(27)"*"CHR$(5)CHR$(144)CHR$(1);
940 FOR J=1 TO 400: LPRINT CHR$(Z(J));:Z(J)=0:  NEXT J
950 LPRINT CHR$(27)"3"CHR$(3)
960 NEXT I
970 LPRINT CHR$(27)"2";
980 FOR I=1 TO 10: LPRINT CHR$(7);: NEXT I
990 LPRINT CHR$(27)CHR$(64);
1000 LPRINT:LPRINT
1010 PRINT"Neue Zeichnung J/N ?"
1020 L$=INKEY$: IF L$="" THEN 1020
1030 IF L$="J" THEN 10
1040 PRINT CHR$(12)
1050 END
1060 LPRINT"Kartesiche Koordinaten von ";A$
1070 LPRINT: FOR I=1 TO N: LPRINT I,B(I,1),B(I,2),B(I,3)
1080 NEXT I: RETURN
1090 LPRINT"Bindungsmatrix von ";A$
1100 LPRINT: FOR I=1 TO M: LPRINT I,BI(I,1);BI(I,2): NEXT I
1110 PRINT"Soll korrigiert werden ?"
1120 L$=INKEY$: IF L$=""THEN 1120
1130 IF L$="N" THEN 380
1140 INPUT"Welche Bindung soll korrigiert werden ";I
1150 INPUT"Neue Bindung i,j ";BI(I,1),BI(I,2)
1160 GOTO 1110
1170 FOR I=1 TO N: AY=B(I,Q): AZ=B(I,P): J=I+N
1180 B(J,Q)=AY*.990268+AZ*.139173*SI
1190 B(J,P)=AZ*.990268-AY*.139173*SI
1200 B(J,T)=B(I,T):NEXT I: RETURN
1210 FOR I=1 TO 200: C1=CH(I)+C1: C2(I)=C1: NEXT I
1220 HH=1: GOTO 700
```

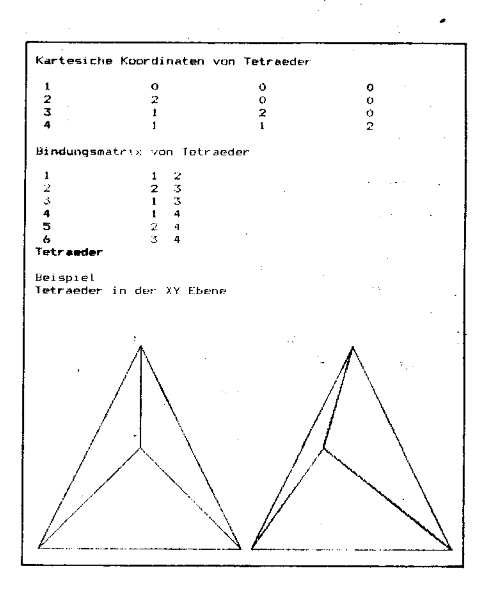


Bild 1 Programmbeispiel

U-880-Editor für Mikrorechner

Andreas Bogatz
Karl-Marx-Universität Leipzig,
Sektion Physik

Sollen die Möglichkeiten eines Mikrorechners bezüglich seines Befehlssatzes, der Peripherieelemente und der maximalen Abarbeitungsgeschwindigkeit voll ausgenutzt werden, so ist es stets notwendig, die Programme statt in einer Hochsprache – etwa C, PASCAL, BASIC ... – direkt in Maschinenkode zu erzeugen.
Zur Unterstützung einer effektiven Programmierung werden dafür Übersetzerprogramme benutzt, die die ‚menschenverständliche' Syntax der Befehlsmnemonik (Quellkode) in die ‚maschinenverständliche' Form (Objektkode) überführen. Zur Vereinfachung der Adreßrechnung bei Sprüngen und Unterprogrammaufrufen ist es dabei nützlich, mit symbolischen Adressen zu arbeiten, d. h. mit Marken (Label).

Üblicherweise werden für kommerzielle Mikrorechnersysteme solche Übersetzerprogramme für die Richtung Quellkode → Objektkode angeboten, wobei die Arbeitsweise blockorientiert ist. Daher muß ein Programm komplett im Quellkode vorliegen und wird in einem durchgehenden Übersetzerlauf in den Objektkode überführt. Diese Arbeitsweise ist während der Programmtestphase relativ umständlich, da nach jeder Quelltextänderung ein erneuter kompletter Übersetzerlauf erfolgen muß.
Häufig bieten diese Programmsysteme auch keine Möglichkeit der Reassemblierung, d. h. der Rückwandlung von fertigen Maschinenprogrammen in Quelltexte.
Eine andere Arbeitsweise bietet dagegen z. B. das Editor-System des MC 80. Dort ist es möglich, Objektkode zu reassemblieren, zu editieren und zu assemblieren, wobei die Übersetzung zeilenorientiert erfolgt. Nach der Eingabe einer Quellzeile erfolgt ein Syntax- und Markentest, bei richtiger Eingabe wird sofort übersetzt. Dieser zeitsparende Dialogbetrieb ist gerade während des Programmtests und der Fehlerbeseitigung nützlich. Nachteilig ist jedoch, daß dabei keine Erzeugung von kommentierten Quelltexten zur Programmdokumentation möglich ist. Außerdem können nur 127 Marken vereinbart werden.
Es bestand daher die Aufgabe, ein Programmsystem zu schaffen, welches die blockorientierte mit der zeilenorientierten Arbeitsweise vereint sowie eine nach Art und Umfang verbesserte Arbeit mit symbolischen Adressen enthält.

Praktische Realisierung

Das Programmsystem besteht aus den Komponenten:

- Quelltexteditor
- Blockübersetzer- und rückübersetzer
- Zeileneditor.

Der Quelltexteditor gestattet das Erzeugen und Bearbeiten von Quelltexten. Der Blockübersetzer und -rückübersetzer dient der Wandlung von Quelltexten in Maschinenkode und umgekehrt und benutzt Unterprogramme aus dem Programm Zeileneditor.
Der Zeileneditor enthält die wesentlichsten Unterprogramme und soll daher etwas näher beschrieben werden.
Der Kern dieses Programmpaketes besteht zunächst aus einer Tabelle, in der die Befehlsmnemonik (ASCII-Ketten) und die entsprechenden Maschinenkodebytes abgelegt sind. Um Tabellenplatz zu sparen, wurden Gemeinsamkeiten der Befehlsmnemonik nur einmal abgelegt (Gruppenmnemonik). Außerdem werden Befehlsspezifika wie z. B. Registernamen, Sprungbedingungen usw. über getrennte Tabellen erschlossen. Eine ähnliche Arbeitsweise liegt auch dem Programmsystem des MC 80 zugrunde.
Tafel 1 veranschaulicht die Organisation der Tabelle.
Die Übersetzungstabelle besteht aus der Befehlsmnemonik, dem zugehörigen Maschinenkode MC und einer Zusatzinformation ZI.
Die Tabelle besitzt folgende Struktur: FFH/Gruppenmnemonik/ZI/MC/Mnemonikfortsetzung/ZI/MC/...
Die Mnemonik wird in ASCII-Ketten abgelegt, wobei Zahlenangaben, häufig auftretende Register, Registergruppen usw. durch Sonderzeichen gekennzeichnet sind und über gesonderte Tabellen erschlossen werden.
Insgesamt benötigt der Tabellenteil weniger als 1 KByte. Weiterhin existieren im Kern des Programmpaketes die Unterprogramme Assembler und Reassembler sowie Unterprogramme zur Verwaltung der Markentabelle. Alle diese Programme wurden bisher als Kern bezeichnet, da sie unabhängig von der konkreten Hardware des verwendeten U-880-Rechners sind. Sie belegen insgesamt etwa 3 KByte. Eine Anpassung an den jeweiligen Rechnertyp ist lediglich beim Programmteil der Bildschirm- und Kommandoverwaltung notwendig. Hierfür existieren derzeit beim Verfasser Varianten für den MC 80 und den Eigenbaurechner KMU 880.
Bild 1 zeigt am Beispiel des Eigenbaurechners KMU 880 die Bildausschrift im Dialogbetrieb (zeilenorientierte Betriebsart).
Der Assembler verarbeitet folgende Zahlendarstellungen:

– Dualzahlen	LD	A,10100011B
– Dezimalzahlen	LD	IX,6712
– Hexazahlen	LD	HL,0FE00H
– symbolische Angaben	CALL	INIT.2
– auch mit Offset	JMP	ANFANG+1E00H
– relative Angaben	JPZ	#-1200

Es können beliebig viele Marken (soweit der Speicher reicht) vereinbart werden, wobei zwischen einem und sechs Zeichen pro Marke zulässig sind. Der Offset bei symbolischen und relativen Angaben kann innerhalb des 64-KByte-Bereiches sowohl dual, hexadezimal oder dezimal angegeben werden. Weiterhin verarbeitet der Assembler Wertzuweisungen eines Bytes oder eines Doppelbytes sowie Zeichenkettendefinitionen mit

Tafel 1 Aufbau der Mnemoniktabelle (Ausschnitt)

Code	Zeichen	Symbol	Bedeutung
21H	!	N1	1-Bytezahl auf Byte 1 (pseudo)
22H	"	NN2	2-Bytezahl auf Byte 1,2 (pseudo)
23H	#	N2	1-Bytezahl auf Byte 2
24H	$	N4	1-Bytezahl auf Byte 4
25H	%	NN23	2-Bytezahl auf Byte 2,3
26H	&	NN34	2-Bytezahl auf Byte 3,4
65H	e	EE	Distanz Byte 2 bei relativem Sprung
64H	d	DD	Distanz Byte 3 bei Indexregister
69H	i	II	IX oder IY
71H	q	R1	Register Bit 012 (Quelle)
7AH	z	R2	Register Bit 345 (Ziel)
72H	r	RR	Doppelregister (Bit 45)
63H	c	CC	Sprungbedingungen Bit 345
62H	b	N	Bitnummer Bit 345

Die Zusatzinformationen sind wie folgt verschlüsselt:

Bit 7 = 1	Kennzeichen Zusatzinformation	
Bit 6 = 0	unbenutzt	
Bit 5 = 0	unbenutzt	
Bit 4 = 0	unbenutzt	
Bit 3210		
0000	kein Vorsatzbyte	(80H)
0001	ED	(81H)
0010	CB	(82H)
0100	DD oder FD	(84H)
0110	DDCB oder FDCB	(86H)
1000	Pseudooperationen	(88H)

```
----MNEMONIKTABELLE-------------
MNT: DM   'LD A,'
     DW   00A80H     ;LD A, (BC)
     DM   '(BC)'
     DW   01A80H     ;LD A, (DE)
     DM   '(DE)'
     DW   03A80H     ;LD A, (NN23)
     DM   '(%)'
     DW   05781H     ;LD A, I
     DM   'I'
     DW   05F81H     ;LD A,R
     DM   'R'
     DB   0FFH       ;TRENNBYTE
     DM   'LD'
     DW   00280H     ;LD (BC), A
     DM   '(BC)' A'
     DW   01280H     ;LD (DE),A
     DM   '(DE)' A
     DW   04781H     ;LD I, A
     DM   'I,A'
     DW   04F81H     ;LD R, A
     DM   'R, A'
     DW   02A80H     ;LD HL, (NN23)
     DM   'HL' (%)'
     DB   0FFH       ;TRENNBYTE
     DM   'LD (id),'
     DW   07084H     ;LD (II+DD), R1
     DM   'q'
     DW   03684H     ;LD (II+DD), N4
     DM   '$'
     DB   0FFH       ;TRENNBYTE
     DM   'LD SP,'
     DW   0F980H     ;LD SP, HL
     DM   'HL'
     DW   0F984H     ;LD SP, II
     DM   'i'
     DB   0FFH       ;TRENNBYTE
     DM   'LD z,'
     DW   04080H     ;LD R2, R1
     DM   ...'q'
```

Tafel 2 Steuerbefehle im Dialogbetrieb

B ack
Anzeige 5 Befehlszeilen zurückschalten
D elet
Streichen bis ausschließlich angegebener Adresse; ohne Angabe: anstehenden Befehl streichen
G o
Anzeige ab 5 Befehlszeilen vor angegebener Adresse; ohne Angabe ab 5 Zeilen vor Programmende (4x0FFH)
H alt
Markierung der angegebenen Adresse als Haltepunkt; ohne Angabe: Haltepunkt am bearbeiteten Befehl
I nsert
Einfügen von Leerbefehlen (NOP) bis ausschließlich angegebener Adresse; ohne Angabe ein Leerbefehl (Voraussetzung: RAM hinter Programm auf benötigter Länge frei)
L ink
Binden des Programmes mit bearbeitetem Befehl auf angegebener Adresse; ohne Angabe Binden auf momentanen Bereich im Speicher
M ove
Verschieben des Programmes und der Marken von Adresse 1 bis ausschließlich Adresse 2 (ohne Angabe von Adresse 2 bis Programmende) auf Bereich ab zu bearbeitender Zeile (Voraussetzung: RAM auf Länge ab Bearbeitungsadresse frei)
N ext
Anzeige 5 Zeilen vorwärts schalten
P rint
Drucken ab Adresse 1 bis ausschließlich Adresse 2; ohne Angabe von Adresse 2 bis Programmende (4x0FFH)
R AM
Anzeige HEXA/ASCII ab angegebener Adresse; ohne Angabe ab Bearbeitungsadresse
T est
Übergang in den Testmodus ab angegebener Adresse; ohne Angabe ab Bearbeitungsadresse

Tafel 3 Übersicht zu den Pseudobefehlen

DB...	– Bytezuweisung
DW...	– Doppelbytezuweisung
DM 'Zeichenkette'	– Zeichenkettendefinition
DS...	– Speicherreservierung angegebener Länge (Eintrag von NOP)
EXT...	– Markenwertzuweisung als Konstante
END	– Ende der Übersetzung
M	– für HL, z. B. LD M, A –→ LD (HL), A
ORG...	– stellt den Programmzähler auf die angegebene Adresse

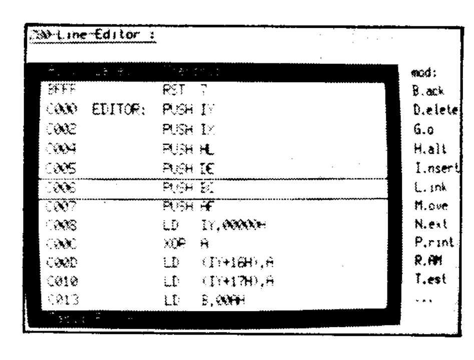


Bild 1 Schirmbild während der Dialogbetriebsart

maximal 30 Zeichen. Verschiedene Pseudobefehle unterstützen den blockorientierten Übersetzerlauf. In der Dialogbetriebsart können Programme erstellt, geändert und verschoben werden.

Ein Testprogramm ermöglicht den Programmtest im Schrittbetrieb oder im Lauf bis zu einem vorgebbaren Haltepunkt. Als Ergänzung kann ein Speicherbereich HEXA/ASCII angezeigt werden, wobei durch einen dynamischen Bildaufbau sich ändernde Zeilen beobachtbar sind.

Im Sinne einer guten Programmübersicht befindet sich die zu bearbeitende Programmzeile in der Bildmitte, wobei der Cursor unter dem ersten Zeichen des Markenfeldes steht. Somit werden 5 Zeilen vor und 6 Zeilen nach der Bearbeitungszeile sichtbar. Die von der Tastatur erzeugten Zeichen werden direkt auf die Bearbeitungszeile geschrieben (auch Tabulation, horizontale Cursorbewegungen und clear) und nach Betätigung der *Enter*-Taste wird die Zeile übersetzt bzw. eine Fehlermeldung ausgegeben. Wird die Bearbeitungszeile jedoch mit vertikalen Cursorbewegungen verlassen, so wird der alte Zustand der Bearbeitungszeile regeneriert und das angezeigte Bild in der entsprechenden Richtung gerollt. Mit Hilfe von Steuerkommandos (ctrl-Taste) können verschiedene Hilfsfunktionen aufgerufen werden; Tafel 2 vermittelt dazu eine Übersicht. Verschiedene der Steuerkommandos fordern Parameter an, die wahlweise als absolute Zahlenangaben, relative Zahlenangaben oder in symbolischer Form (Marke, eventuell mit Offset) angebbar sind.

Abschließend zeigt Tafel 3 die möglichen Pseudobefehle, die auch den blockorientierten Übersetzerlauf unterstützen.

EPROM-fähige TURBO-PASCAL-Programme

Durch den Einsatz moderner hochintegrierter Schaltkreise lassen sich bei geringem Platz- und Energiebedarf relativ hohe Speicherkapazitäten realisieren. Damit wird es möglich, auch die Systemsoftware von Geräten des wissenschaftlichen Gerätebaus ganz oder teilweise in höheren Programmiersprachen zu erstellen. International hat sich vor allem C bei den 16-Bit-Prozessoren für diesen Zweck durchgesetzt. Aus folgenden Gründen erschien die Verwendung von TURBO-PASCAL in unserer Einrichtung sinnvoll:

- ausgezeichnete Programmierumgebung
- einheitliche Programmiersprachen für System und Anwendersoftware
- genügende Code- und Laufzeiteffizienz der compilierten Programme
- problemlose Einbindung von Assemblerroutinen
- flexible Möglichkeiten zur Programmänderung
- vorhandene Programmiererfahrungen.

Da die unter SCPX lauffähige Variante von TURBO-PASCAL (Bürocomputer, Personalcomputer) Arbeitszellen im Bereich von 0000H ... 00FFH (Grundseite) benutzt, wurde das Laufzeitmodul (etwa 8 KByte) so verändert, daß diese Datenbereiche auf eine beliebige andere Adresse verschoben werden können. In Verbindung mit der CHN-Compileroption des TURBO-PASCAL-Compilers entstehen EPROM-fähige Programme, in denen auch die READ- und WRITE-Prozeduren möglich sind. Die Speicheraufteilung für ein solches Programm kann dann z. B. so aussehen:

EPROM
Turbo-Laufzeitmodul
Gerätedriver (Code)
Anwenderprogramme (Code)

RAM
Anwenderprogramme (Daten)
Gerätedriver (Daten)
Grundseite

Der eigentliche TURBO-PASCAL-Compiler bleibt unverändert. Die Gerätedriver oder auch ein komplettes Betriebssystem können ebenfalls ganz oder teilweise in TURBO-PASCAL geschrieben werden. Abhängig von der konkreten Problemstellung benötigen die Programme im Verhältnis zu Assemblerprogrammen erfahrener Programmierer 2- bis 4mal mehr Speicher. Bei Aufgabenstellungen, die besonders viele arithmetische Funktionen beinhalten, liegt das Verhältnis besser. Die unbestrittenen Vorteile der Verwendung einer Hochsprache hinsichtlich Lesbarkeit, Änderungsfreundlichkeit und Portabilität wiegen diesen Nachteil aber bei weitem wieder auf.

Neben der erwähnten Änderung des Laufzeitmoduls liegt außerdem ein PASCAL-Programm vor, das die Erzeugung von INLINE-Code aus der PRN-Datei des ASM (Makroassembler unter SCPX) gestattet. Optional kann der Quelltext des Assemblerprogramms als Kommentar mit in die INLINE-Anweisung aufgenommen werden. Externe Marken erscheinen mit ihrem Namen im Inline-Code. Zur Testunterstützung wurde außerdem eine Möglichkeit der schrittweisen Abarbeitung der TURBO-PASCAL-Programme unter Ausnutzung der U-Compilerdirektive erarbeitet. *Dr. B. Petzold*

☒ KONTAKT ☎

Forschungszentrum für Bodenfruchtbarkeit Müncheberg, Abt. wissenschaftlicher Gerätebau, Wilhelm-Pieck-Str. 72, Müncheberg, 1278; Tel. 82241

BASIC-Sprachübersicht für KC85/3, KC87 und SCP-BASIC-Interpreter BASI

Klaus-Dieter Kirves, Mühlhausen

Anweisung	Bemerkungen	KC85/3	KC87	SCP-BASIC
ABS(X)	berechnet absoluten Betrag	Y=ABS(X)	Y=ABS(X)	Y=ABS(X)
AND	logische UND-Funktion	Z=X AND Y	Z=X AND Y	Z= X AND Y
ASC(X$)	ASCII-Code des ersten Zeichens	Y=ASC(X$)	Y=ASC(X$)	Y=ASC(X$)
AT	positioniert PRINT-Anweisung	AT(Z,S)	AT(Z,S)	-
	auf Zeile,Spalte			
ATN(X)	Arcustangensberechnung im	Y=ATN(X)	Y=ATN(X)	Y=ATN(X)
	Bogenmaß			
AUTO	automat. Zeilennumerierung	AUTO[a][,s]	AUTO[a][,s]	AUTO[a][,s]
BEEP	gibt akustisches Signal	BEEP N	BEEP N	-
BLOAD	lädt Maschinenprogramm	BLOAD	-	-
BORDER	Bildschirmrandfarbe	-	BORDER N	-
BYE	Rückkehr zum Betriebssystem	BYE	BYE	(SYSTEM)
CALL	Aufruf Maschinenprogramm	CALL adr	CALL adr	CALL adr,[parameter]
	Adresse dezimal			
CALL*	Aufruf Maschinenprogramm	CALL*adr	CALL*adr	-
	Adresse hexadezimal			
CDBL	wandelt einfach genaue Zahl	-	-	Y=CDBL(X)
	in doppelt genaue Zahl			
CHAIN	Nachladen von Programmsegmenten	-	-	CHAIN[MERGE]"NAME",[Zeilen]
CHR$	liefert Zeichen mit Code von X	A$=CHR$(X)	A$=CHR$(X)	A$=CHR$(X)
CINT	erzeugt Integer-Variable	-	-	Y=CINT(X)
CIRCLE	zeichnet Kreis um X,Y	CIRCLE X,Y,R,[F]	-	-
	mit Radius R in Farbe F	*)		

1

CLEAR	löscht Variablenspeicher	CLEAR[I[,E]]	CLEAR[I[,E]]	CLEAR[E[,S]]
CLOAD	lädt Programm von Kassette	CLOAD "NAME"	CLOAD "NAME"	-
CLOAD*	lädt Datenfeld von Kassette	CLOAD*"NAME";	CLOAD*"NAME";	-
		FELDVar.	FELDVar	-
CLOSE	schliesst Datei	CLOSE#n *)	-	CLOSE[#]Nummer[,...]
CLS	löscht Bildschirm	CLS	CLS	-
COLOR	stellt Farbe ein	COLOR VC,H]	-	-
	(auch als PRINT-Erweiterung)			
COMMON	Übergabe von Informationen	-	-	COMMON Var...
	zwischen Programmsegmenten			
CONT	setzt Programm fort	CONT	CONT	CONT
CSAVE	speichert Programm auf Kassette	CSAVE"NAME"	CSAVE"NAME"	-
COS(X)	berechnet die Cosinusfunktion	Y=COS(X)	Y=COS(X)	Y=COS(X)
CSAVE*	speichert Daten auf Kassette	CSAVE*"NAME";	CSAVE"NAME ;	(PRINT#)
		FELDVar.	FELDVar.	
CSGN	wandelt doppelt genaue Zahl	-	-	Y=CSGN(X)
	in einfach genaue Zahl			
CSRLIN	ermittelt Zeilennummer des	A=CSRLIN(N)	-	-
	Cursors	*)		
CVD	erzeugt doppelt genaue Zahl	-	-	Y$=CVD(X)
	aus Zeichenkette			
CVI	erzeugt Integerzahl	-	-	Y$=CVI(X)
	aus Zeichenkette			
CVS	erzeugt einfach genaue Zahl	-	-	Y$=CVS(X)
	aus Zeichenkette			
DATA	vereinbart Konstantenliste	DATA C1[,C2...]	DATA C1[,C2...]	DATA C1[,C2...]
DEEK	liest zwei Speicherplätze	X=DEEK(N)	X=DEEK(N)	-
DEF FN	definiert eine Funktion	DEF FNY(P)=Aus.	DEF FNY(P)=Aus.	DEF FN(P[,Pn...])=Aus.
DEFDBL	vereinbart doppelt genaue Var.	-	-	DEFDBL Var[,Var...]
DEFINT	vereinbart Integervariable	-	-	DEFINT Var[,Var...]

2

DEFSGN	vereinbart einfach genaue Var.	-	-	DEFSGN Var[,Var...]
DEFSTR	vereinbart Stringvariable	-	-	DEFSTR Var[,Var...]
DEFUSR	trägt Startadresse	-	-	DEFUSRN=Adresse
	für USER-Funktion ein			
DELETE	löscht Zeilen	DELETE A,E	DELETE A,E	DELETE A,E
DIM	legt Grosse eines Variablen-	DIM FELDNAME	DIM FELDNAME	DIM FELDNAME(I[,J...])
	feldes fest	(I[,J...]	(I[,J...]	(I[,J...]
DOKE	beschreibt zwei Speicherzellen	DOKE A,D	DOKE A,D	-
EDIT	ruft Zeileneditor auf	EDITn	EDITn	EDIT n
ELSE	Alternative bei IF	ELSE Anw.	ELSE Anw.	ELSE Anw.
END	beendet Programmabarbeitung	END	END	END
EOF	zeigt Dateiende an	-	-	Y=EOF(Dateinr.)
EQV	liefert Äquivalenzfunktion	-	-	Z=X EQV Y
ERASE	gibt dimensionierte Felder frei	-	-	ERASE Feldn.[,Feldn...]
ERL	enthält Zeilennummer bei ERROR	-	-	IF ERL=N THEN...
ERR	enthält Fehlercode	-	-	IF ERR=n THEN...
ERROR	simuliert Programmfehler	-	-	ERROR(Fehlercode)
EXP(X)	liefert Exponentialfunktion	Y=EXP(X)	Y=EXP(X)	Y=EXP(X)
FIELD	legt Satzstruktur fest	-	-	FIELD [#]Nr.,
				Länge AS Var.[,...]
FILES	zeigt Dateiverzeichnis an	-	-	FILES ["Dateibez."]
FIX	liefert den ganzzahligen Teil	-	-	Y=FIX(X)
	einer einfach genauen Zahl			
FN	ruft definierte Funktion auf	Z=FNY(P)	Z=FNY(P)	Y=FNY(P[,Pn...])
FOR	legt Programmschleifenanzahl	FOR Var=A TO E	FOR Var=A TO E	FOR Var=A TO E
	fest			
FRE	zeigt verfügbaren Speicher an	Y=FRE(Var)	Y=FRE(Var)	Y=FRE(Var)
GET	liest Satz in den Puffer	-	-	GET[#]Dateinr.[,Satznr.]
GOSUB	ruft Unterprogramm	GOSUB n	GOSUB n	GOSUB n
GOTO	springt unbedingt	GOTO n	GOTO n	GOTO n

3

HEX$	wandelt Zahl in Hexadezimalzahl	-	-	Y$=HEX$(X)
IF	bedingte Sprung- oder	IF...THEN	IF...THEN	IF...THEN
	Handlungsanweisung	IF...GOTOn	IF...GOTOn	IF...GOTO n
IMP	logische Implikationsfunktion	-	-	Z= Y IMP X
INK	stellt Vordergrundfarbe ein	INK N	INK N	-
	(auch als PRINT-Erweiterung)			
INKEY$	fragt Tastatur ab	A$=INKEY$	A$=INKEY$	A$=INKEY$
INP	fragt Port ab	A=INP(I)	A=INP(I)	A=INP(I)
INPUT	wartet auf Eingabe	INPUT Var	INPUT Var	INPUT Var
INPUT#	liest Daten ein	INPUT#n Var	INPUT#n Var	INPUT#Dateinr.,Var
INPUT$	erwartet N Tastatureingaben	-	-	A$=INPUT$(N)
INSTR	sucht eine Zeichenkette in	A=INSTR(A$,B$)	A=INSTR(A$,B$)	A=INSTR(A$,B$)
	einer anderen	*)		
INT	wandelt Variablenwert in	A=INT(X)	A=INT(X)	A=INT(X)
	Integerwert			
JOYST	fragt Spielhebel ab	(vorbereitet *)	JOYST(N)	-
KEY	belegt Funktionstaste	KEY N *)	-	-
KEYLIST	listet Funktionstastenbelegung	KEYLIST *)	-	-
KILL	löscht Datei	-	-	KILL "Dateibez."
LEFT$	bildet Teilzeichenkette	A$=LEFT$(B$,N)	A$=LEFT$(B$,N)	A$=LEFT$(B$,N)
LEN	liefert die Länge einer	A=LEN(B$)	A=LEN(B$)	A=LEN(B$)
	Zeichenkette			
LET	weist Variable Wert zu	LET B=Aus.	LET B=Aus.	LET B=Aus.
	(kann entfallen)			
LINE	zeichnet Linie	LINE XA,YA,XE,	-	-
		YE[,F] *)		
LINES	legt Zeilenanzahl beim Listen	LINES N	LINES N	-
	fest			
LINE	liest Zeile ein	-	-	LINE INPUT Var
INPUT				

4

LINE	liest Zeile von Datei ein	-	-	LINE INPUT# Dateinr.
INPUT#				,Var
LIST	listet Programm aus	LIST [n]	LIST [n]	LIST [n],[m]
LIST#	listet auf Gerät aus	LIST#N"NAME"	LIST#N"NAME"	(LLIST)
LLIST	druckt Programm aus	(LIST#N)	(LIST#N)	LLIST[N],[M]
LOAD	lädt Programm	LOAD#N"NAME"	LOAD#N"NAME"	LOAD"Dateibez."[,R]
LOC	übergibt Satznummer	-	-	A=LOC(Dateinr.)
LOCATE	setzt die Cursorposition	LOCATE Z,S	-	-
LOF	zeigt Anzahl der Sätze an	-	-	A=LOF(Dateinr.)
LOG	berechnet den natürlichen	(LN(X))	(LN(X))	Y=LOG(X)
	Logarithmus			
LN	berechnet den natürlichen	Y=LN(X)	Y=LN(X)	(LOG(X))
	Logarithmus			
LPOS	zeigt Position des Druckkopfes	-	-	A=LPOS(Dummy)
LPRINT	gibt PRINT-Anweisung an Drucker	(PRINT#n)	(PRINT#n)	LPRINT Var...
LPRINT	druckt formatiert	-	-	PRINT USING "Form.";Var...
USING				
LSET	trägt Zeichenkette in Satz ein	-	-	LSET A$=X$
MERGE	mischt Programmteile	-	-	MERGE"Datei"
MID$	erzeugt Teilzeichenkette	A$=MID$(B$,P,N)	A$=MID$(B$,P,N)	A$=MID$(B$,P,N)
MKD$	wandelt doppelt genaue Zahl	-	-	A$=MKD$(X)
	in 8-Byte-Zeichenkette			
MKI$	wandelt Integerzahl	-	-	A$=MKI$(X)
	in 2-Byte-Zeichenkette			
MKS$	wandelt doppelt genaue Zahl	-	-	A$=MKS$(X)
	in 4-Byte-Zeichenkette			
MOD	liefert Rest bei Division	-	-	Z=Y MOD X
NAME	benennt Dateien um	-	-	NAME "neu" AS "alt"
NEW	löscht Programm	NEW	NEW	NEW
NEXT	schliesst FOR-NEXT-Schleife ab	NEXT[X[,Y...]]	NEXT[X[,Y...]]	NEXT [X[,Y...]]

5

NOT	Negationsfunktion	A=NOT(B)	A=NOT(B)	A=NOT (B)
NULL	legt Dummyzeichen fest	NULL N	NULL N	-
OCT$	wandelt Zahl in Oktalwert um	-	-	B$=OCT$(A)
ON GOTO	bedingte Mehrfachverzweigung	ON Var	ON Var	ON Var
		GOTOn[,m...]	GOTOn[,m...]	GOTO n[,m...]
ON	setzt fort bei Fehler	-	-	ON ERROR Reak.
ERROR				
ON	bedingter Mehrfach-	ON Var	ON Var	ON Var
GOSUB	unterprogrammaufruf	GOSUBn[,m...]	GOSUBn[,m...]	GOSUB n[,m...]
OPEN	eröffnet Datei	OPEN#n"NAME" *)	-	OPEN "x",#Dateinr,"Datei"
OPTION	legt Indexgrenze für Felder	-	-	OPTION BASE N
BASE	fest			
OR	logische ODER-Funktion	Z=X ORY	Z=X ORY	Z=X OR Y
OUT	gibt Daten an Port	OUT P,D	OUT P,D	OUT P,D
PAPER	stellt Hintergrundfarbe ein	PAPERN	PAPERN	-
	(auch als PRINT-Erweiterung)			
PAUSE	wartet N*0,1s	PAUSE N	PAUSE N	-
PI	Konstane 3.14159	PI	PI	
PEEK	liest Byte von Speicherplatz	A=PEEK(I)	A=PEEK(I)	A=PEEK(I)
POKE	schreibt Byte auf Speicherplatz	POKE I,B	POKE I,B	POKE I,B
POS	gibt Cursorposit. in der Zeile	A=POS(Dummy)	A=POS(Dummy)	A=POS(Dummy)
PRESET	löscht Bildpunkt	PRESETX,Y	-	-
PRINT	gibt auf Bildschirm aus	PRINT[Erw.;]Var	PRINT[Erw.;]Var	PRINT[Erw.;]Var...
		auch ?	auch ?	auch ?
PRINT#	gibt auf Gerät aus	PRINT#n...	PRINT#n...	PRINT#n...
PRINT	gibt formatiert aus	-	-	PRINT USING "Form.";Var...
USING				
PSET	setzt Bildpunkt	PSETX,Y[,F]	-	-
PTEST	testet Bildpunkt	A=PTEST(X) *)	-	-
PUT	schreibt Satz in Datei	-	-	PUT[#] Dateinr.Satznr.

6

RANDOMIZE	startet den Zufallsgenerator	RANDOMIZE *)	-	RANDOMIZE
READ	liest Daten aus DATA-Zeilen	READ Var...	READ Var...	READ Var...
REM	kennzeichnet Kommentar	REM...	REM...	REM...
		oder !	oder !	oder !
RENUMBER	numeriert Programmzeilen neu	RENUMBER[a[,e	RENUMBER[a[,e	RENUM[n[,a[,S]]]
		[,n[,S]]]]	[,n[,S]]]]	
RESET	schliesst alle Dateien ab	-	-	RESET
RESTORE	setzt DATA-Zeiger auf Zeile	RESTORE n	RESTORE n	RESTORE n
RESUME	schliesst Fehlerbehandlung ab	-	-	RESUME Fort.
RETURN	bewirkt Rückkehr aus Unter-	RETURN	RETURN	RETURN
	programm			
RIGTH$	bildet Teilzeichenkette	A$=RIGHT$(B$,N)	A$=RIGHT$(B$,N)	A$=RIGHT$(B$,N)
RND	berechnet Pseudozufallszahl	A=RND(S)	A=RND(S)	A=RND(S)
RSET	trägt Zeichenkette in Satz ein	-	-	RSET A$=B$
RUN	startet Programm	RUN[n]	RUN[n]	RUN [n]
				RUN "NAME"[,R]
SAVE	rettet Programm	(CSAVE)	(CSAVE)	SAVE "NAME"[,Op.]
SGN	bestimmt Signumfunktion	A=SGN(X)	A=SGN(X)	A=SGN(X)
SIN	berechnet Sinusfunktion	Y=SIN(X)	Y=SIN(X)	Y=SIN(X)
SOUND	erzeugt Töne	SOUNDT1,V1,T2,		
		V2[,L[,D]]		
SPACE$	erzeugt Zeichenkette aus N	-	-	A$=SPACE$(N)
	Leerzeichen			
SPC	erzeugt N Leerstellen bei	SPC(N)	SPC(N)	SPC(N)
	PRINT-Anweisung			
SQR	berechnet Quadratwurzel	Y=SQR(X)	Y=SQR(X)	Y=SQR(X)
STEP	bestimmt Schrittweite bei	STEP Aus.	STEP Aus.	STEP Aus.
	FOR-NEXT-Schleife			
STOP	unterbricht Programmabarbeitung	STOP	STOP	STOP
STR$	wandelt Zahl in Zeichenkette	A$=STR$(X)	A$=STR$(X)	A$=STR$(X)

7

STRING$	vervielfacht Zeichenkette	STRING$(N,B$)	STRING$(N,B$)	STRING$(N,B$)
SWAP	tauscht Variablen	-	-	SWAP A,B
SWITCH	schaltet Module	SWITCH A,S *)	-	-
SYSTEM	kehrt zum Betriebssystem zurück	(BYE)	(BYE)	SYSTEM
TAB	tabuliert bei Print-Anweisung	TAB(N)	TAB(N)	TAB(N)
TAN	berechnet Tangensfunktion	Y=TAN(X)	Y=TAN(X)	Y=TAN(X)
TROFF	schaltet Zeilennummeranz. aus	TROFF	TROFF	TROFF
TRON	schaltet Zeilennummeranz. ein	TRON	TRON	TRON
USR	ruft Funktion auf	Y=USR(X)	Y=USR(X)	Y=USR(X[,Var...])
VAL	bestimmt numerischen Wert einer	A=VAL(X$)	A=VAL(X$)	A=VAL(X$)
	Zeichenkette			
VARPTR	bestimmt Anfangsadresse einer	-	-	A=VARPTR(X)
	Variable oder des I/O-Puffers			A=VARPTR#n
VGET$	liest Zeichen vom Bildschirm	A$=VGET$ *)	-	-
VPEEK	liest Byte aus Bildwieder-	A=VPEEK(I)	-	-
	holspeicher			
VPOKE	schreibt Byte in Bildwiederhol-	VPOKE I,A	-	-
	speicher			
WAIT	wartet auf Reaktion am Port	WAIT I,J[,K]	WAIT I,J[,K]	WAIT I,J[,K]
WHILE	baut Iterationsschleife auf	-	-	WHILE Bed. Reak. WEND
WEND				
WIDTH	legt Ausgabezeilenlänge fest	WIDTH N	WIDTH N	WIDTH [LPRINT] N
WINDOW	stellt Bildschirmfenster ein	WINDOW[ZA,ZE	WINDOW[ZA,ZE	-
		,SA,SE]	,SA,SE]	
WRITE	gibt alternativ zu PRINT aus	-	-	WRITE Var...

8

Anmerkungen:
*) nicht bei Kassettenversion für KC85/2
() adäquate Anweisung
[] optional
n Zeilennummer

Anw. Anweisung
Aus. Ausdruck
Bed. Bedingung
Erw. Erweiterung
Form. Formatanweisung

Fort. Fortsetzungsanweisung
Op. Option
Reak. Reaktion
Var. Variable
Var. . . .Variablenliste

COMPUTER · CLUB · MP

Liebe Redaktion!

Wir sind ebenso wie andere Computerclubs der Auffassung, daß Hardware- und Software-Lösungen unter den Clubs kostenlos getauscht werden sollten.
Unser Computer-Club hat in der Arbeitsgruppe „Z1013-Eigene Technik" Mitglieder aus der gesamten DDR. Wir möchten Euch bitten, in zwangloser Folge Lösungsvorschläge zur Diskussion zu stellen.
In einem ersten Beitrag bieten wir unterschiedliche Varianten einer Taktfrequenzumschaltung des MRB Z1013.
Weitere Vorschläge für Veröffentlichungen unsererseits sind:
– Speichererweiterung 32K, 48K, 64K
– Anschluß von ALPHA-Tastaturen.

Hans-Jochen Bachmann
Computer-Club robotron, Leipzig

Wir greifen Eure Anregung gerne auf, da sie den Interessen eines breiten Leserkreises entgegenkommt, und veröffentlichen nachfolgend den ersten Beitrag.

Elektronische Taktfrequenzumschaltung beim Z1013

Programme, die mit 1 MHz Taktfrequenz auf Kassette abgespeichert wurden, werden von der 2-MHz-Variante nicht eingelesen und umgekehrt. Aus diesen Gründen wurden für den Programmaustausch zwischen beiden Varianten Taktfrequenzumschaltungen als Hardware-Lösungen realisiert.
Diese Taktfrequenzumschaltungen ermöglichen im Betriebsmodus des Z1013 das Umsetzen von Programmen in die jeweils andere Taktfrequenz. Die Umschaltung wurde so gewählt, daß der Takt zu jeder Zeit vorhanden bleibt und damit der Refresh-Zyklus der dynamischen RAMs aufrecht erhalten wird.
Die Schaltungen wurden auf Lochraster-Leiterplatten realisiert und mit Abstandsbolzen aus Plast etwa 2 cm über der Grundleiterplatte des MRB Z 1013 aufgestockt (Bild 1).

Schaltungsbeschreibung

Variante 1

Diese Variante stellt eine Minimalvariante dar und wurde nur für Umspeicherungen in die jeweils andere Taktfrequenz konzipiert. Die Umschaltung erfolgt manuell über einen

5P
360 1k 1k
B1
2MHz
1MHz
TAKT
A1
S1
A1 : D 150
B1 : VQA 13
S1 : DIL-Schalter

Bild 2 Variante 1

Schalter (DIL-Schalter) im Grundzustand des Mikrorechners. Dabei wird die Torfunktion der IS D 150 zur Auswahl der gewünschten Taktfrequenz genutzt. Die Auswahl wird durch den an Pin 4 gelegten Pegel (H oder L) bestimmt. Die LED signalisiert den 2-MHz-Betriebszustand.

Variante 2

Dieser Lösungsvorschlag wurde für die Steuerung der Taktfrequenzumschaltung von der Tastatur (Taste!) bzw. für eine softwaremäßige Steuerung über einen PIO-Kanal entwickelt. Auch hierbei wird die Torfunk-



Bild 3 Variante 2

Bild 4 Variante 3

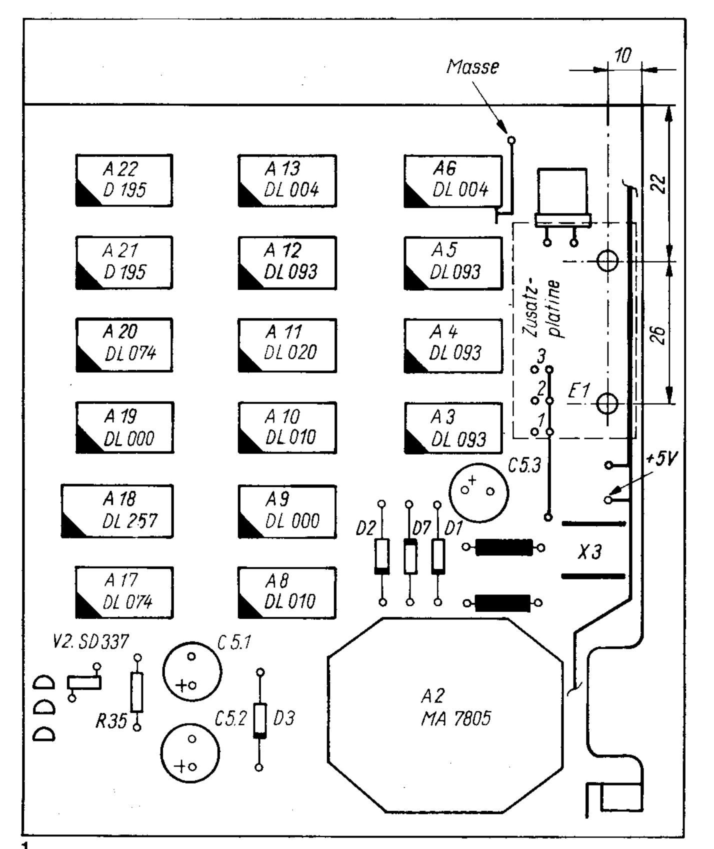
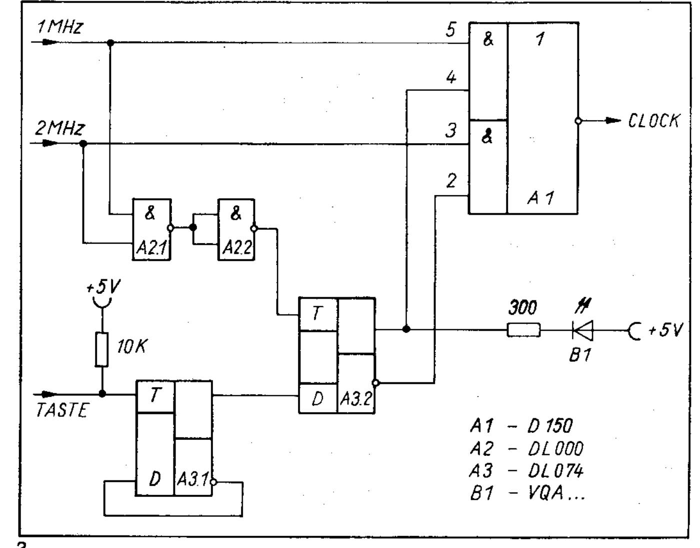
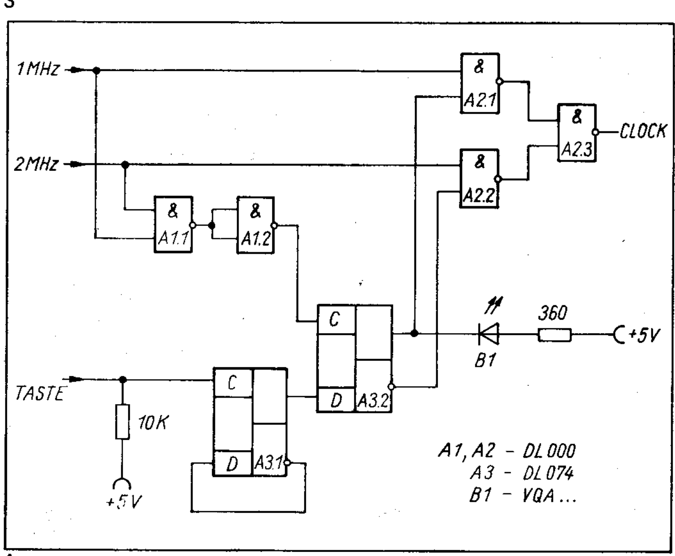


Bild 1 Geometrische Anordnung





tion der IS D 150 genutzt. Die Umschaltung erfolgt nur zu eindeutig definierten Taktzeiten.

Die Bedingungen sind:
1 MHz = H
2 MHz = H
IS 3.2 = H/L Flanke an Eingang D

Damit ist die Umschaltung zu jedem beliebigen Zeitpunkt, auch in laufenden Programmen, möglich.
Dies ist z. B. vorteilhaft nutzbar bei Tonausgaben in Programmen. Für die Steuerung über die Tastatur ist die gesamte Schaltung zu realisieren, während bei einer Software-Steuerung A 3.1 entfallen kann und der PIO-Kanal direkt an den Eingang D der A 3.2 geführt wird. Bei Anschluß einer Alpha-Tastatur kann z. B. bei der Tastatur K 7659 die LED C99 zur Anzeige des Betriebsmodus genutzt werden.

Variante 3

Dieser Lösungsvorschlag erfolgte mit Gattern statt mit der IS D 150 und erfüllt schaltungstechnisch die gleichen Bedingungen wie in Variante 2 beschrieben.

Umrüstung des MRB Z 1013

Eine Änderung der Original-Hardware ist nicht notwendig, es sind lediglich 2 zusätzliche Bohrungen von 3,4 mm Durchmesser (Bild 1) für die Befestigung der Plastabstandsbolzen mit 2 Plastschrauben M3 erforderlich sowie die zum Anschluß erforderliche Verdrahtung.
Auf der Originalplatine werden die Taktbrücke E1 ausgelötet und die 3 Drahtverbindungen 2 MHz, 1 MHz und Takt angelötet. Außerdem wird die Spannungsversorgung +5 V und Masse von den in Bild 1 gekennzeichneten Punkten verdrahtet.
Dabei ist auf folgende Anschlußbedingungen der IS D 150 zu achten:

Pin 3 – 2 MHz
Pin 5 – 1 MHz

Der Schalter bzw. die Taste zur Steuerung der Umschaltung kann an beliebiger Stelle angeordnet werden.
Nach dem Verdrahten erfolgt die Befestigung der bestückten Zusatzplatine. Werden die Taste und die LED auf der Tastatur angeordnet, kann der Anschluß über die freien Kontakte B11 und A13 des Steckverbinders x2 erfolgen. *H.-J. Bachmann*

Zu einem Ergebnis des Rundfunk-BASIC-Lehrgangs

In einer Sendung wurde ein BASIC-Programm zum Berechnen von Primzahlzwillingen vorgestellt. Das sind zwei Primzahlen, die den Abstand 2 besitzen, also z. B. 5, 7 oder 71, 73. Es war danach gefragt, wie das vorgestellte Programm funktioniert und wie es schneller zu machen geht. Die Antworten waren erfreulich, erstaunlich und es wurden Programme eingesandt, die bis zu einer rund 10fachen Geschwindigkeit gelangten. Doch nicht nur dies ist erstaunlich, nein auch, mit welcher Akribie hier vorgegangen wurde und wie vielfältig die Möglichkeiten sind, das eine Problem zu lösen.

```
10 CLS: PRINT TAB(9);"## PRIZWI2 ##": Y=5
20 FOR Z=7 TO 1E6 STEP 2
30 FOR T=3 TO SQR(Z) STEP 2: W=Z/T
40 IF W=INT(W) THEN T=Z: NEXT: NEXT: END
50 NEXT:IF Z=Y+2 THEN PRINT Y,Z: Z=Y+4: NEXT: END
60 Y=Z: NEXT
```

```
10 CLS: PRINT"PRIZBEC"
20 FOR Z=11 TO 1E6 STEP 6: V=1: Z1=Z
30 FOR T=5 TO SQR(Z1) STEP 2:W=Z1/T
40 IF W=INT(W) THEN T=Z1: NEXT: NEXT: END
50 NEXT: IF V=1 THEN V=2: Z1=Z1+2: GOTO 30
60 PRINT Z, Z1: NEXT: NEXT: END
```

```
10 CLS: PRINT TAB(9);"PRIZST": N=2
20 FOR F=5 TO 1E6 STEP 6
30 FOR G=5 TO SQR(F+N) STEP 6
40 H=F/G: IF H=INT(H) THEN NEXT F
50 I=(F+N)/G: IF I=INT(I) THEN NEXT F
60 J=(F+N)/(G+N): IF J=INT(J) THEN NEXT F
70 K=F/(G+N): IF K=INT(K) THEN NEXT F
80 NEXT: PRINT F,F+N: NEXT
>
```

```
10 CLS: PRINT TAB(9);"## PRIZ4 ##":DIM A(600)
20 FOR Z=17 TO 1000000 STEP 600
30 FOR M=7 TO SQR(Z+600) STEP 2
40 FOR N=INT(Z/M+0.999)*M TO Z+600 STEP M
50 A(N-Z)=Z
60 NEXT
70 M=M+4
80 FOR N=INT(Z/M+0.999)*M TO Z+600 STEP M
90 A(N-Z)=Z
100 NEXT
110 NEXT
120 FOR P=Z TO Z+590 STEP 30
130 FOR M=P TO P+24 STEP 12
140 IF A(M-Z)=Z THEN NEXT: NEXT: NEXT: END
150 IF A(M-Z+2)=Z THEN NEXT: NEXT: NEXT: END
160 PRINT M,M+2: NEXT: NEXT: NEXT: END
```

```
10 CLS: DIM A(3),M(3)
20 A(0)=7: A(1)=9: A(2)=11: A(3)=13: GOTO 60
30 FOR K=3 TO SQR(A(I)) STEP 2: W=A(I)/K
40 IF W=INT(W) THEN M(I)=0: RETURN
50 NEXT: M(I)=1: RETURN
60 FOR I=1 TO 2: GOSUB 30: NEXT
70 IF M(1)*M(2)=1 THEN PRINT A(1),A(2): GOTO 120
80 IF M(1)=1 THEN I=0:GOSUB 30
90 IF M(0)=1 THEN PRINT A(0),A(1): M(0)=0: GOTO 120
100 IF M(2)=1 THEN I=3: GOSUB 30
110 IF M(3)=1 THEN PRINT A(2),A(3): M(3)=0
120 FOR I=0 TO 3: A(I)=A(I)+10: NEXT: GOTO 60
```

Hier seien 4 besonders interessante Lösungen im Vergleich zum Original abgedruckt.

PRIZWI2	Original	1
PRIZBEC	F. Bäcker, Bralitz	2*
PRIZST	P. Sturm, Genthin	3*
PRIZ4	Th. Kunert, Dessau	9*
Ohne	D. Bauke, Gera	1.4*

Die letzte Zahl gibt den Faktor der ungefähren Erhöhung der Geschwindigkeit wieder. Man beachte auch die Länge und Lesbarkeit der Programme. *Prof. Dr. H. Völz*

Speichererweiterungsmodul am KC 87

Im KC 87 kommen die gleichen Speichererweiterungsmodule wie im KC 85/1 zum Einsatz, d. h. für ein Erweitern des Arbeitsspeichers des Grundgerätes der RAM-Erweiterungsmodul 690 003.5 (16 KByte) und für eine Festwertspeichererweiterung der ROM-Erweiterungsmodul 690 002.7 (10 KByte).
Die 64 KByte Gesamtspeicherumfang des KC 87 (KC 85/1) sind in 4 Blöcke von je 16 KByte gerastert, ihre Anfangsadressen lauten 0000H (RAM-Bereich des Grundgerätes), 4000H, 8000H, C000H. Diese Anfangsadressen sind in bekannter Weise mit Schaltern auf der jeweiligen Modul-Leiterplatte einstellbar. Beim Einsatz von Speichererweiterungsmodulen ist folgendes zu beachten.

RAM-Erweiterung

Der KC 87 (85/1) erkennt nur einen zusammenhängenden RAM-Bereich. Deshalb wird ein einzeln stehender RAM-Erweiterungsmodul mit der Anfangsadresse 8000H nicht akzeptiert. Wenn ein 32 KByte-RAM-Bereich benötigt wird, ist das nur durch einen RAM-Erweiterungsmodul mit Anfangsadresse 4000H realisierbar.

ROM-Erweiterung

Ein ROM-Erweiterungsmodul umfaßt einen Adreßbereich von 10 KByte ab eingestellter Anfangsadresse. Er darf zur Vermeidung von Doppelbelegungen keinen durch RAM-Module bereits belegten Adreßbereich benutzen, ansonsten unterliegen die Anfangsadresse 4000H und/oder 8000H keiner Einschränkung.
Für die Anfangsadresse C000H ist zu beachten, daß hier normalerweise der BASIC-Interpreter beginnt. Soll in dessen Bereich ein anderes ROM-Programm arbeiten, muß der BASIC-Interpreter eliminiert werden.
Beim KC 85/1 wird dafür der BASIC-Modul gegen den ROM-Erweiterungsmodul getauscht (natürlich bei abgeschaltetem Gerät). Beim KC 87 muß der im Grundgerät integrierte BASIC-Interpreter abgeschaltet werden. Das geschieht durch eine Masseverbindung zwischen den Steckverbinderpunkten X1 : 1B (= Masse) und X1 : 9B (/ROMDI). Die Masseverbindung ist als Lötbrücke serienmäßig ab 1987 in die ROM-Erweiterungsmodule eingebaut. Im KC 85/1 führt diese Masseverbindung allerdings auch zur Abschaltung des Betriebssystems. Demzufolge sind beim Einsatz von Kleincomputern und ROM-Erweiterungsmodulen verschiedener Generationen die Hinweise in der Tafel zu beachten. Diese Hinweise gelten auch für alle aus dem ROM-Erweiterungsmodul abgeleiteten Module wie

IDAS-Modul	690 020.3
EDITOR/ASSEMBLER-MODUL	690 022.8 und
PLOTTER-GRAFIK-MODUL	690 033.2,

die im Adreßbereich C000H – E7FFH angeordnet sind. *D. Lauter*

Kombination		Maßnahmen	
KC-Typ	ROM-Modul mit Seriennummer	ROM-Bereich ab C000H	ROM-Bereich ab 4000H oder 8000H
85/1	unter 870 000	BASIC-Modul ziehen	keine
85/1	ab 870 000	Lötbrücke auf Modul-Lp trennen	
87	unter 870 000	Verbindung zwischen Steckverbinderstift 1B und 9B auf Modul-Lp einfügen	keine
87	ab 870 000	keine	Lötbrücke auf Modul-Lp trennen

COMPUTER MP CLUB

Probleme, Tricks und kurze Routinen

In der Praxis der Rechentechnik erweisen sich immer wieder kleine Erkenntnisse als besonders nützlich. Wir beabsichtigen daher, beginnend in diesem Heft unter der Rubrik Computerclub einige spezielle Spalten für derartige Probleme einzurichten. Hier sollen auf der Basis der Einsendungen unserer Leser solche Lösungen regelmäßig abgedruckt werden. Sie sollten aus diesem Grund sehr kurz dargestellt werden und der Umfang in der Regel eine Schreibmaschinenseite nicht übersteigen. Entsprechend dem Profil unserer Zeitschrift müssen die dargestellten Fakten das Gebiet der Mikrorechentechnik betreffen. Es interessieren Hard- und Software-Fragen.

Die der Redaktion zugesandten Probleme können Anfragen an andere Leser sein, wie z. B.:

- Routinen zur einfachen Erzeugung von Kreis- bzw. Ellipsenbögen
- Formatierung von Zahlen in BASIC
- Interface-Probleme bezüglich verschiedener Rechner.

Unter Tricks verstehen wir Lösungen, die durch ihre Einfachheit und Leistungsfähigkeit bestechen. Hierzu können z. B. gehören:

- Kassettenroutinen mit Namenseingabe, z. B. CSAVE*""+N$; A (siehe Heft 6/87, S. 162)
- RUN im Programm, um Felder zu redefinieren.

Kurze Routinen müssen eine relativ universelle Verwendung garantieren, z. B.:

- die speicher- und ladbare Bildschirmkopie in diesem Heft
- das Problem der Primzahlzwillinge in diesem Heft.

Wir bitten Sie, liebe Leser, uns im allgemeinen Interesse Ihre Beiträge hierzu zu übersenden. Sie nützen damit vielen anderen.

Speichern von Bildschirminhalten beim KC 85/2 und /3

Der KC 85/2 bzw. /3 besitzt eine hochauflösende Grafik. Die Bilder benötigen zum Aufbau oft sehr lange Zeit. Daher wäre es bequem, wenn man die fertigen Bilder auf Kassette archivieren könnte. Infolge der Blockanzeige beim Retten und Laden sowie der Interaktivität Tastatur-Bildschirm geht dies jedoch nicht ohne weiteres. Hier wird eine anders geartete Methode vorgestellt. Sie verwendet im BASIC-Programm zwei Maschinenroutinen. Die erste, bei 406H, doppelt den Bildschirminhalt (Pixel-RAM) auf den Adreßbereich 1800 – 3FFFH. Die zweite legt ihn von dort auf den Pixel-RAM zurück. Hier sind Zusatzmöglichkeiten zur Manipulation ergänzt.

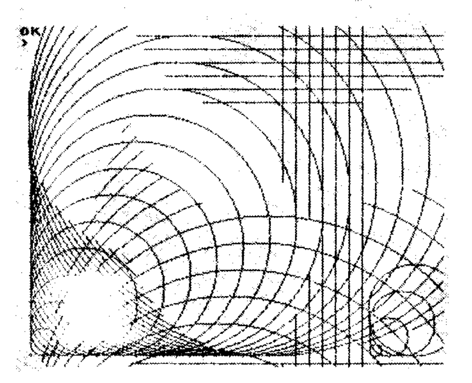
Ferner können Sie mittels Cursor und Eingabe zusätzliche Kennzeichen auf dem Bild anbringen. Für eine Bildschirmhardcopy wird der Cursor vorher automatisch unterdrückt.

Anordnung des Hilfsprogramms

Um diese Möglichkeiten zu erproben, wurde ein kurzes BASIC-Programm geschrieben, in dessen Zeile 0 die beiden Maschinenroutinen eingebunden sind. Den HEX-Dump des gesamten Programms zeigt Bild 1. Es ist infolge seiner Kürze mühelos mit MODIFY einzugeben und dann zu retten. Die Arbeit damit gestaltet sich wie folgt: Einschalten des Rechners, BASIC normal initialisieren (Modul-BASIC beim KC 85/2 oder KC 85/3 direkt); mit BYE zurück ins Betriebssystem; Routine laden; MODIFY 3D7 mit: 3D7: B004 REBASIC. Wenn Sie jetzt LIST ausführen, erscheinen zunächst auf dem Bildschirm

```
0400 00 5A 04 00 00 8E F5 C5
0408 D5 E5 DB 88 CB D7 D3 88
0410 21 FF 7F 11 FF 17 01 01
0418 28 ED B0 E1 D1 C1 F1 C9
0420 7F 7F 43 4F 50 59 01 45
0428 21 FF 17 11 FF 7F 7E A8
0430 12 23 13 7A FE A8 20 F6
0438 CD 03 F0 17 C9 0C 0A 0D
0440 43 6F 70 79 72 69 67 68
0448 74 20 48 2E 20 56 6F 65
0450 6C 7A 20 31 39 38 37 0D
0458 0A 00 78 04 0A 00 A1 20
0460 33 30 2C 36 31 30 30 3A
0468 20 DD 20 30 2C 33 31 2C
0470 30 2C 33 39 3A 20 99 00
0478 80 04 14 00 81 20 58 B4
0480 31 20 A6 20 33 30 30 20
0488 A0 20 31 30 00 A2 04 1E
0490 00 F3 20 58 AC 37 2C 20
0498 58 AC 37 2C 20 58 2C 20
04A0 37 00 AF 04 28 00 82 3A
04A8 20 90 AE 34 30 36 00 00
04B0 00 00 00 00 00 00 00 00
```

```
Copyright H. Voelz 1987
10 CLEAR 30,6100: WINDOW 0,31,0,39: CLS
20 FOR X=1 TO 300 STEP 10
30 CIRCLE X+7, X+7, X, 7
40 NEXT: CALL*406
OK
>
```



unübliche Zeichen. Dann wird der Bildschirm gelöscht und schließlich folgt Bild 2. Ein RUN muß dann Bild 3 erzeugen.

Danach liegt der Bildschirminhalt auch auf dem Adreßbereich 1800H – 3FFFH. Hiervon können Sie sich wie folgt überzeugen: CLS, BYE, MENU und mit %COPY 0 ist sofort das Bild wieder vorhanden. Mit %COPY FF erhalten Sie ein vollständig negiertes Bild, und mit %COPY 77 eine andere interessante Struktur.

Funktionieren der Routine

Mit CLEAR 30,6100 wird der Stringraum auf 30 Byte begrenzt und die obere Adresse auf 6100D = 17D4H begrenzt. Dadurch wird der BASIC-Raum auf etwa 5 KByte reduziert. Dies reicht noch in vielen Fällen. Ansonsten ist ein zusätzlicher 16-oder 64-K-RAM zu verwenden und der zweite Wert bei CLEAR entsprechend zu erhöhen. Der erste Wert ist zu erhöhen, wenn Strings verwendet werden. Die Zeilen 30 bis 40 beschreiben den Bildschirm mit dem Muster von Bild 3, und CALL* 406 legt ihn danach auch auf den Speicherbereich 1800 bis 3FFFH. Dies erfolgt einfach mit dem LDIR-Befehl. Doch zuvor müssen HL, DE, BC gesetzt sowie der Bildschirm angeschaltet werden, letzteres erfolgt mit:

DB 88 IN A, (88H); von PIO
CB D7 SET 2, A
D3 88 OUT (88H), A

Es sei noch erwähnt, daß jeweils ein Byte mehr übertragen wird. Dies ist notwendig, damit im Maschinencode keine 0 vorkommt. Die Wiedergaberoutine arbeitet ähnlich, jedoch wird dabei der Inhalt mit

XOR B

verändert. Deshalb sind nach COPY die verschiedenen Werte notwendig.

Retten und Laden

Nach dem bisher Gesagten ist es leicht möglich, den Bildschirminhalt zu retten und erneut in den Rechner zu laden. Dies geschieht im Betriebssystem mit SAVE 1800 3FFF bzw. LOAD des geretteten Files. Danach kann wie oben weiter verfahren werden.

Es ist auch kein Problem, Bildinhalte an andere Adreßbereiche zu verlagern oder gar mehrere zugleich abzulegen. In diesem Fall sind nur die Byte bei den Adressen 414/415 H und 429/42 AH entsprechend zu verändern.

Prof. Dr. H. Völz

Nutzung der Tape-LED

Die Tape-LED des KC 85/3 (/2) ist am Datenbit 5 des PIO-Ports mit der Adresse 88H angeschlossen. Soll dieses Datenbit verändert werden, muß gewährleistet sein, daß alle anderen Datenbits unverändert bleiben. Näheres in /1/.

Das folgende kleine BASIC-Programm läßt die LED blinken:

```
10 OUT 136,INP(136) OR 32:PAUSE 2:! LED on
20 OUT 136,INP(136) AND 223:PAUSE 2:! LED off
30 GOTO 10
```

K.-D. Kirves

Literatur

/1/ –;Kleinkomputer KC 85/3 System-Handbuch; VEB Mikroelektronik „W. Pieck“ Mühlhausen

Kleincomputer-Teachware

Für die Lehrlings-, Schüler- und Studentenausbildung bieten wir eine nachnutzbare Teachware unter Nutzung des Kleincomputers KC 85/3 an. Das *Lehrbuch* (Autor Dr. Peter Kolbe) umfaßt 287 Seiten (mit 143 Seiten für Aufgaben, Lösungen, Programme und Übungen), 19 Abbildungen, 2 Tabellen und 78 Struktogramme zu den 38 Beispielen des Programmpakets PEGASUS.
Es beschreibt die wesentlichen Hardwarekomponenten eines Mikrorechnersystems und erläutert sie anhand einer Minimalkonfiguration mit dem KC 85/3. Weiterhin werden die Grundzüge der Softwareentwicklung (Prinzipien, Konzepte und Phasen) vorgestellt. Wesentlich ist die Einführung von Linear-Struktogrammen als zweckmäßiges, grafisches Beschreibungsmittel für den Softwareentwurf.
Der Hauptanteil des Lehrbuches wird durch 38 grafische Aufgabenstellungen bestimmt mit Lösungsentwürfen anhand von 78 Struktogrammen sowie BASIC-Implementationen auf der *Programmkassette*. Die Programme sind im Programmpaket PEGASUS (Programmlösungen elementarer grafischer Aufgabenstellungen und Strukturerzeugungen) zusammengefaßt.
Hochschule für industrielle Formgestaltung, Wissenschaftsbereich Designmethodik, Neuwerk 7, Halle, 4020; Tel. 85 03 73

Prof. Dr. sc. Frick

CPASET

CPASET ermöglicht den Austausch von Disketten, die im Home-Format (800 KByte) des Betriebssystems CPA formatiert sind, zwischen den Personalcomputern A 7100 (SCP) und PC 1715 (CPA). CPASET läuft am A 7100 unter SCP 1700 und ermöglicht für alle Laufwerke (A–D) vom Typ MFS 1.6 das Einstellen des CPA-Home-Formates:

Sektorierung: DS×80×5×1024
Systemspuren: keine
Verzeichnis: 192 Einträge
Spur-0-Behandlung: keine
Diskettenkapazität: 800 KByte.

Ein mit CPASET umgestelltes Laufwerk erlaubt das Schreiben und Lesen derartiger Disketten am A 7100.
Voraussetzung für die Anwendung von CPASET ist ein vorheriger Lauf des im SCP 1700 enthaltenen Dienstprogramms DISKSET zur Initialisierung des Floppy-Controllers auf die 5×1024-Sektorierung, da CPASET selbst keine Controllerinitialisierung durchführt.
Komponenten:
CPASET.CMD Maschinenprogramm (SCP 1700, CP/M-86)
CPASET.DOC Dokumentation
VEB Blechwarenfabrik Oberoderwitz, Dr. Volker Sieber, Dorfstr. 146, Oberoderwitz, 8716; Tel. Niederoderwitz 7459

Hentschel

Grafikfähige IFSS-Schnittstelle am K 6311

Um die serielle Schnittstelle IFSS unter Beibehaltung der IFSS-Interfacekassette grafikfähig zu gestalten (K 6311 ist serienmäßig nur mit V.24 und Centronics grafikfähig), wurde die Umprogrammierung der EPROM-Kassette mit folgenden Parametern vorgenommen:

- Übertragungsrate: 9600 Bd
- Übertragungsart: Duplex
- Übertragungsbreite: 7 Datenbits
- Sicherung: ungerade Parität
- Steuerungsprotokoll: XON = DC 1, XOFF = DC 3

Der Datenaustausch erfolgt nach den Bedingungen der V.24 – Schnittstelle mit folgenden Einschränkungen:

- Anzahl der Teilbilddaten (Einzelbildmodus) begrenzt infolge der Übertragungsbreite von 7 Bit auf 127 Schritte
- Ansteuerung der Drucknadel 1 nicht möglich (für zusammenhängende Bilder werden nur 6 Nadeln benötigt)
- Steuercodesequenzen *Sende Gerätestatus* und *Sende Geräteerkennung* auf der Empfangsschleife und *Fehlerstatus DC 4* auf der Sendeschleife sind nicht auswertbar.

Die Programme zur Hardcopy des Grafikbildschirms VIS 2A über die Schnittstelle IFSS sind in den Betriebssystemen SCP und EIEX implementiert und mit dem Mikrocomputersystem MC 80.31 getestet worden. Umprogrammierungen der Interfacekassetten IFSS werden in unserer Einrichtung vorgenommen.
Karl-Marx-Universität Leipzig, Bereich Medizin, Abt. Mikrorechentechnik, Liebigstr. 27, Leipzig, 7010; Tel. 7 167385

Dr. Börner

Kurvenorientierte Grafiksoftware

Im Rahmen der Entwicklung eines intelligenten Meßsystems wurden von uns Programme zur 2dimensionalen Kurvendarstellung auf dem Plotter K 6418 und den Druckern K 6311/K 6312 (Einzelpunktmodus) erstellt (Programme für K 6313/K 6314 sind in Vorbereitung).
Alle Programme wurden unter dem Betriebssystem SCP mit der Programmiersprache PASCAL entwickelt und sind für einen Arbeitsspeicherbereich von 48 KByte ausgelegt. Sie sind für die Darstellung von gemessenen oder berechneten Kurvenverläufen konzipiert. Je nach Programmversion bestehen Wahlmöglichkeiten zwischen Netz- und Achsendarstellung, Mehrkurven- und Mehrdiagrammdarstellung, unterschiedlichen Linientypen, verschiedenen Formaten, Eingabe oder Berechnung der Markenabstände.
Hochschule für Verkehrswesen Dresden, Technikum Diagnostik und Zuverlässigkeit, Postschließfach 103, Dresden, 8072; Tel. 4660158

Holst

Leiterkartenherstellung

Im Zusammenhang mit dem Aufbau von Kapazitäten zur Herstellung von durchkontaktierten Leiterkarten (DLK) im VEB Datenverarbeitungszentrum Rostock wurde eine Reihe von spezifischen Ausrüstungen projektiert. Das sind z. B.:

- Lackzieheinrichtung
- Galvanikanlage für DLK.

Für diese Spezialausrüstungen liegen Konstruktionsunterlagen vor, die von Interessenten im Rahmen der Nachnutzung erworben werden können.
VEB Datenverarbeitungszentrum Rostock, Betriebsteil EPMR, Abt. Absatz, E.-Schlesinger-Str. 37, Rostock 6, 2500; Tel. 8511

Bruhn

Digitalisierung von Leiterplatten-Konstruktionsunterlagen

Das Programmpaket dient der rechentechnischen Bearbeitung und maschinellen Zeichnung von Leiterplattenkonstruktionsunterlagen.
Merkmale:
- Kopplung zwischen HDG 1/HDG 2 und BC A 5120/A 5130 unter SIOS
- Wandlung grafischer Daten einer Leiterplattenvorlage in die NC-Sprache AZT-80 und Anlegen der Satzdatei auf Diskette
- Eigene Makrodefinition und Einbindung in die Satzdatei
- Manipulieren mit der Satzdatei
- Erzeugen einer variablen Schrift
- Ausgabe von Steuerdaten für ALZG 3, LM-Z 10 und LKA
- Rechentechnische Ableitung weiterer Leiterplattenunterlagen (z. B. Lötmaskenaufdruck, Bearbeitungszeichnung, Koordinatenlisten außer Raster befindlicher Bohrungen).

VEB Zentrum Wissenschaft und Technik Dresden, Betrieb des VEB Kombinat Rundfunk und Fernsehen, PF 969, Dresden, 8060

Mittel- und langfristige Planung – Erneuerungspaß (MLP-EP)

Das Projekt MLP-EP ist ein Zweiebenenprojekt zur Speicherung und Auswertung von Erneuerungspaßdaten auf ESER-Rechner und zur Übernahme und Auswertung ausgewählter Teildatenbestände auf Personalcomputern PC 1715. Der Gesamtdatenbestand aller Erneuerungspässe wird in einer Datenbank gespeichert und verwaltet, die mit dem Datenbankbetriebssystem DBS/R (Version 5.2) organisiert wird.
Informationsauszüge aus dieser Datenbank können am PC 1715 arbeitsplatzbezogen ausgewertet werden. Die Auswertung erfolgt unter Nutzung von REDABAS, einschließlich des Listgenerators.
Für die Datenerfassung am PC 1715 oder BC A 5120/A 5130 wird auf die Anwendung des Programmes ERZPA der SZS orientiert. Hardwarebasis:
EC 1055/1056
PC 1715 mit 2 MFS 1.6 und 8″-Beistellgefäß
EC 5075 (Diskettenlesegerät für 8″-Disketten)
Softwarebasis:
OS/ES 6.19, SVS; DBS/R 5.2; UNI 3; SCP 1715; REDABAS.
VEB Robotron-Elektronik Dresden, Grunauer Str. 2, Dresden, 8012

Faktografisches Informationsrecherchesystem

REDAFAKT basiert auf REDABAS und dient der Erfassung,

Pflege, Recherche und wahlfreien Ausgabe von Daten fester Länge auf Bildschirm und/oder Drucker. Test und Erprobung erfolgen auf A 5120/A 5130 und PC 1715.
Entsprechend dem Erfassungsmodus erfolgt die Dateneingabe auf Maskenbasis nach dem Prinzip Merkmalsname: Merkmalsaussage (n).
Das Programm ist überall dort nutzbar, wo sich der dokumentierende Sachverhalt in das o. g. Erfassungsschema gliedern läßt, der Datenbestand sich qualitativ für die dezentrale Rechentechnik und ihre spezifischen Speichermedien eignet sowie die angebotenen Aufbereitungsformen ausreichend erscheinen.
VEB Robotron-Elektronik Dresden, Leitstelle für Information und Dokumentation, PSF 330, Dresden, 8012

Zentrale Datenbank Raum- und Flächennutzung

Die Rationalisierungslösung besteht aus dem Organisationsteil (Org.-Anweisung) und der Programmdokumentation.
Der Org.-Teil regelt die Verantwortlichkeiten bei der Planung und Vergabe von Räumen und Flächen sowie der Organisation des Zusammenwirkens zwischen den Partnern.
In der Programmdokumentation sind alle Hauptproduktions-, Hilfsproduktions-, Nebenproduktions-, Lagerflächen, Büro- und Verwaltungseinrichtungen sowie alle gemeinschaftlich genutzten Einrichtungen gespeichert.
Genutzt werden PC oder BC mit 2 Laufwerken unter SCP und REDABAS.
VEB Halbleiterwerk Frankfurt (O), PSF 379, Frankfurt (O), 1200

Terminkontrolle im Leitungsprozeß

Das Programm wurde für den PC 1715 in REDABAS geschrieben und wird zur Kontrolle vorgegebener Termine im Betrieb eingesetzt.
Über eine Menüführung ist es dem Bediener möglich, Termine neu aufzunehmen, zu löschen bzw. zu ändern. Die Termine können auf dem Bildschirm aufgelistet bzw. gedruckt werden. Es können bis zu 3500 Termine auf eine Diskette gebracht werden. Ein Diskettenwechsel ist vorgesehen. Der Vorteil des Systems besteht darin, daß sich jährlich wiederholende Termine übertragen lassen und eine Programmerweiterung ohne größeren Aufwand realisierbar ist.
VEB Funkwerk Kölleda, Eugen-Richter-Str., PSF 48, Kölleda, 5234

EPROM-Programmier-Steckeinheit für BC A 5120/A 5130

Mit dem Programm können EPROMs mit der Speicherkapazität 1, 2, 4, 8, 16 KByte gelesen und programmiert werden (2708, 2716, 2732, 2764, 27 128, K573RF1, K573RF2).
Voraussetzung: Bürocomputer mit Programmiersteckeinheit
Anwenderspeicher: 30 KByte RAM (Mindestgröße)
Kontaktierung der EPROMs im spannungslosen Zustand!
Funktionen: Lesen, Vergleichen, EDC-Berechnung, Einfügen eines Bytes, Löschen eines Bytes, Inhalt ändern, Programmieren, Teilprogrammierung, Füllbyte, Systemrückkehr.
VEB-Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt, Annaberger Straße 93, Karl-Marx-Stadt, 9048

Stücklistengewinnung STL 1.5

Mit Hilfe der Programme kann der Konstrukteur Stücklisten gewinnen sowie Artikelrecherchen, Masse- und Oberflächenberechnungen durchführen (mit selbständiger Eintragung in Listen), indem Kurzbezeichnung der Bauelemente und Positionsnummern eingegeben werden.
Vereinfachter Dateiaufbau ist möglich.
Als Grundlage dient ein BC mit 3 Folienspeichern oder ein PC 1715. Betriebssystem ist SCP oder CP/A; Voraussetzung sind Bauelementedateien.
VEB Studiotechnik Berlin, Rungestr. 25/27, Berlin, 1020

Wir suchen ...

... eine Lösung zum Anschluß von LBE (K 6200) bzw. Lochbandstanzer/Lochbandleser an einen PC 1715 oder BC A 5120.
VEB MAW Armaturenwerk Prenzlau, Abt. TVF, Wilhelm-Pieck-Straße 91, Prenzlau, 2130; Tel. 32459 oder 32250
Brausendorf

3. UNIX-Problemseminar

Vom 21. bis 24. April 1987 führte das Rechenzentrum des Informatikzentrums der TU Dresden in Nassau/Erzgeb. ein Problemseminar zur Anwendung UNIX-kompatibler Betriebssysteme durch. Es nahmen 45 Entwickler und Anwender aus 21 Einrichtungen des MHF, der Industrieministerien und der NVA teil.

Schwerpunktmäßig wurden folgende Themen behandelt:

1. UNIX-kompatible Systeme für ESER
2. Anwendung und Eigenschaften des Systems BSD 2.9 für 16-Bit-SKR-Technik
3. UNIX auf Arbeitsplatz- und Personalcomputern
4. Gekoppelte UNIX-Systeme.

Zum ersten Schwerpunkt berichteten die Vertreter der TU Karl-Marx-Stadt, der TH Leipzig und der Friedrich-Schiller-Universität Jena über die Bereitstellung von weiteren Treibern für die Geräte der ESER-Peripherie und darüber, daß grundlegende Voraussetzungen für eine Weiterentwicklung des Systems geschaffen werden konnten. Erste Erfahrungen mit dem Betrieb der Stand-alone-Version des VMX an der TU Karl-Marx-Stadt zeigten eine Durchsatzerhöhung von ca. 60%.
Von Vertretern der Martin-Luther-Universität Halle und des Informatikzentrums der TU Dresden wurde über erste positive Erfahrungen mit dem System BSD 2.9 informiert, das für die 16-Bit-SKR-Technik seit kurzem zur Verfügung steht. Durch Änderungen am Systemkern und am Filesystem wurde die Geschwindigkeit des Systems erhöht. Außerdem ist der Prozeß der Systemgenerierung wesentlich vereinfacht worden. Für den Anwender wird es besonders durch die Verfügbarkeit von Compilern für FORTRAN77, PASCAL und durch ein LISP-System attraktiv.
Im dritten Schwerpunkt wurden die Systeme WEGA und MUTOS 1700 des P8000 bzw. des A7100 vorgestellt. Die Teilnehmer schätzten ein, daß die Attraktivität UNIX-kompatibler Systeme in dieser Leistungsklasse in entscheidendem Maße von der Bereitstellung peripherer Direktzugriffsspeicher mit einer Kapazität größer 20 MByte abhängt.
In Diskussionen äußerten die Teilnehmer die Beobachtung, daß eine verstärkte „Verkommerzialisierung" von UNIX zu verzeichnen ist. Das zeigt sich u. a.

– in Entwicklung und Vertrieb durch führende Softwarehersteller
– im Schutz der Urheberrechte durch Nichtweitergabe von Systemquellen
– im Schutz des Systems gegen Kopieren
– im preislich gestaffelten Angebot für verschiedene Systemkomponenten.

Es wurde eingeschätzt, daß UNIX gegenüber anderen Betriebssystemen im Angebot anwendungsorientierter Software z. Z. noch zurückbleibt. Hier kann erwartet werden, daß die führenden Softwarehersteller zukünftig ihre Produkte auch für UNIX anbieten. Benötigt werden:

– Programmiersprachen TURBO-PASCAL, MODULA2, COBOL
– Datenbanksysteme
– Graphikpakete zur Unterstützung von Konstruktionsaufgaben.

Um mit dem in der DDR verfügbaren Potential den internationalen Weg mitgehen zu können, ist kooperatives Wirken der Entwickler und Anwender UNIX-kompatibler Systeme notwendig. Dazu hat mit Sicherheit dieses Seminar, nicht zuletzt durch die angenehme Atmosphäre am Tagungsort, einen wichtigen Beitrag geleistet. Es ist beabsichtigt, im März 1988 ein weiteres Seminar auszurichten. Interessenten wenden sich bitte an den Veranstalter.

Dr. C. Kofer

PERSCOMP'87

Die „Nationale Konferenz mit internationaler Beteiligung zur Personal-Computertechnik" der VRB, veranstaltete vom 21. bis 23. April dieses Jahres in Sofia, hat sich bereits mit ihrer zweiten Tagung zu einer respektablen und repräsentativen Veranstaltung dieses hochaktuellen Gebietes von Wissenschaft, Technik und Volkswirtschaft entwickelt. Die PERSCOMP-Konferenzen stehen unter dem Protektorat des Staatsratsvorsitzenden der VRB, Todor Shivkov, und werden von insgesamt 8 Institutionen und Organisationen veranstaltet. Den Vorsitz der Konferenz übernahm der Akademiepräsident der bulgarischen Akademie der Wissenschaften (BAN), Prof. A. Balevski. Die Hauptträgerschaft obliegt dem Institut für Kybernetik und Robotik (ITKR) der BAN Sofia (Dir. Prof. A. Angelov). Die Gestaltung der Tagung sowie der Ablauf können als beispielhaft angesehen werden. Sehr vorteilhaft erwies sich die Einordnung der Plenarsitzungen täglich von 11.00 – 13.00 Uhr mit jeweils 3 bis 4 Übersichts- und Orientierungsthemen vorzugsweise internationalen Charakters (darunter 7 Vorträge von DDR-Teilnehmern).
Das Ziel der Konferenz besteht darin, die Teilnehmer über den national und international erreichten Stand des Entwurfes, der Produktion und vor allem der Nutzung von Personalcomputern in unterschiedlichsten Bereichen der Gesellschaft zu informieren. Zu diesem Zweck war mit der Konferenz eine Ausstellung von Personalcomputern der VRB, mit Geräten aus der DDR und besonders den USA verbunden.
In den Übersichtsthemen dominierten 16- und 16/32-Bit-Personalcomputer (PC-XT und PC-AT) ausschließlich mit der IBM-Portabilität (MS-DOS, PC-DOS), die als defacto-Weltstandard mit langer Lebensdauer ausgewiesen wurden (bis ca. 1995). Die dominierenden Basis-Prozessoren sind der i80286 und der M 68020. Als aufwärtskompatibler 32-Bit-Personalcomputer wird der mit dem i80386 verbesserte PC-AT folgerichtig eingeordnet. Als stark vorwärtsweisende Alternativen dazu erscheinen 32-Bit-PC auf RISC-Prozessor-Basis bzw. die MICRO-VAX. Damit schließen Mikrorechner vollständig zum Systemniveau von Minirechnern auf und übernehmen mit diesen das klassische Minirechner-Niveau vollständig. Entsprechend wurde eine Langlebigkeit der beiden Betriebssystemklassen MS-DOS und UNIX/VMS festgestellt. Wesentlich dabei erscheint die konsequente Öffnung von Entwicklungs-Betriebssystemen (DOS, UNIX) zu Echtzeit-Steuerbetriebssystemen für Prozeßrechner-Funktionen. Dieser internationalen Linie ordnet sich der RGW ein.
Die wichtigsten Themen der Tagungsbeiträge in etwa 30 Sitzungen waren:

– PCs in lokalen Netzen (LAN)
– Peripherie-Geräte für PCs
– PCs in industriellen Anwendungen zur Meßwertverarbeitung und Steuerung technologischer und fertigungstechnischer Prozesse
– CAD/CAM/CAE-Systeme für den rechnergestützten Entwurf, die rechnergestützte Produktionsvorbereitung und -steuerung
– Labormeßwertverarbeitung, Laborautomatisierung
– PCs in der Signalanalyse und Signalsynthese
– Roboter-Steuerungssysteme.

Eine bedeutende Gruppe von Vorträgen befaßte sich mit der Nutzung von Verfahren der künstlichen Intelligenz, mit sozialen Fragen der „Computerisierung der Volkswirtschaft" und Ausbildungs-/Qualifizierungsproblemen.
Die nächste Tagung ist für April 1989 angekündigt.

Prof. Dr. M. Roth

Termine

Lehrgänge des Informatik-Zentrums an der Technischen Universität Dresden für das 2. Halbjahr 1987

Einführung in die Parallelverarbeitung mit Mikroprozessoren
26.11.–27.11.1987
Leitung: Dr.-Ing. Stange
Wissenschaftsbereich Rechnersysteme[1]

Projektierung von lokalen Rechnernetzen
8.12.–11.12.1987
Leitung: Prof. Dr. sc. oec. Garbe
Wissenschaftsbereich Rechnersysteme[1]

Systemprogrammiersprache C
(mit Prüfung; Voraussetzung: Grundlehrgang 1)
14.12.–18.12.1987
Leitung: Doz. Dr. sc. techn. Horn
Bereich Applikation und Service[2]

Teilnahmemeldungen werden erbeten an
[1] Informatikzentrum des Hochschulwesens an der TU Dresden, Stellv. Bereich Weiterbildung (Tel. 4575279) Mommsenstraße 13, Dresden, 8027
[2] Kammer der Technik, BV Dresden, Abt. Weiterbildung (Tel. 2326210) Basteistraße 5, Dresden, 8020

Prof. Dr. Giesecke

Interaktive Bildverarbeitungssysteme für Forschung und Industrie A 6470

robotron

Jahrelange gemeinsame Forschungs- und Entwicklungsarbeit hochqualifizierter Wissenschaftler und Ingenieure führten zu Systemlösungen, die neue Möglichkeiten der Bildverarbeitung bieten.

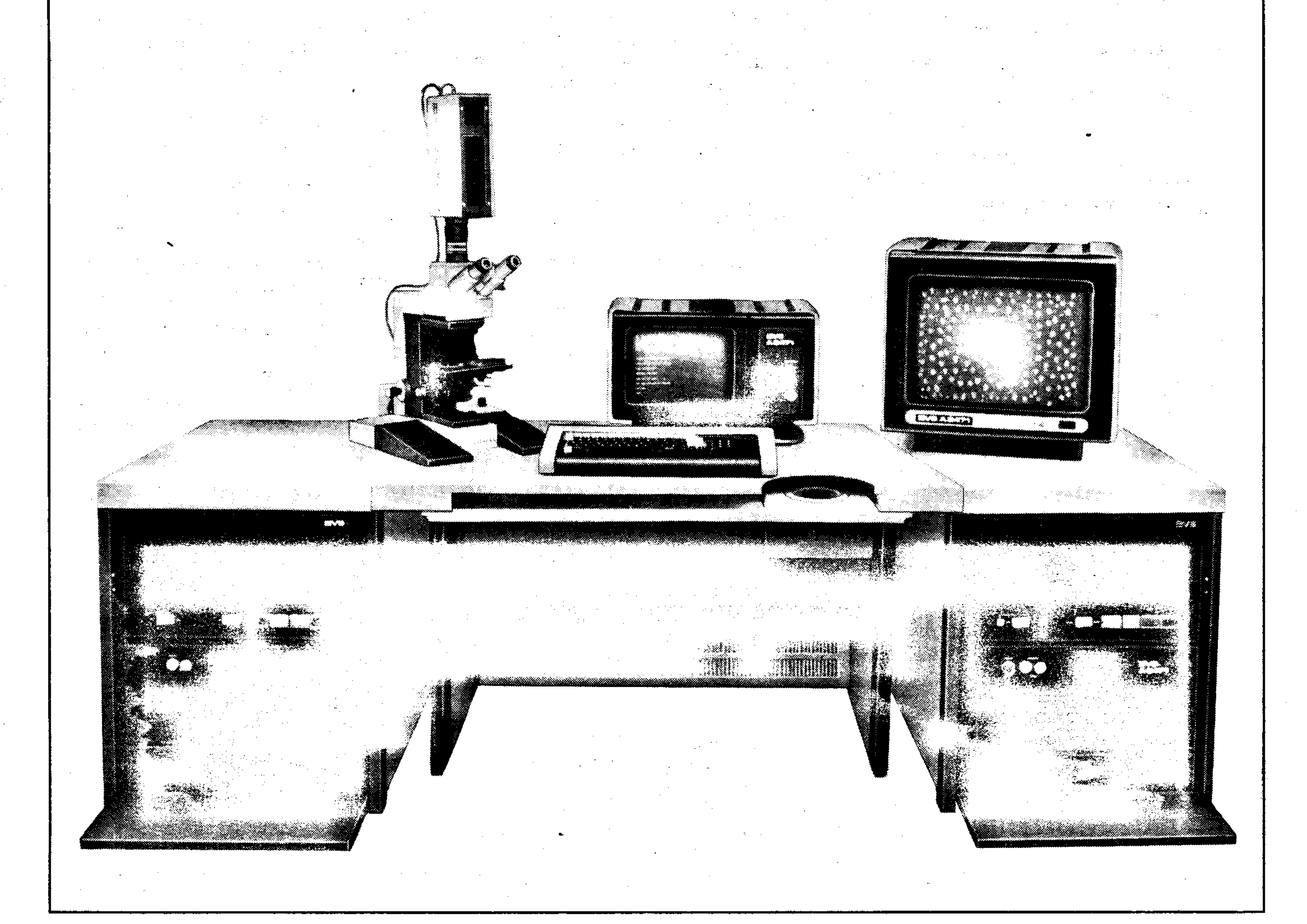
Dazu gehören:
- Automatisierte Mikroskopbildanalyse
- Bildverarbeitungstechnik im industriellen Einsatz
- Fernerkundung der Erde

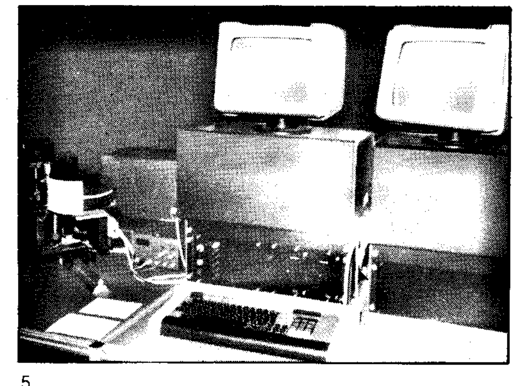
Mit dem interaktiven digitalen Bildverarbeitungssystem robotron A 6470 steht ein modular ausbaufähiges System zur Verfügung, das problemlos den unterschiedlichsten Anforderungen der multispektralen, multitemporalen und lokalen Bildverarbeitung, von der experimentellen Verarbeitung kleiner Bildmengen (BVS A 6471) bis zur Routineverarbeitung mit höchsten Verarbeitungsgeschwindigkeiten (BVS A 6472 und 6473) angepaßt werden kann.
Die wesentlichsten Bildverarbeitungskomponenten, Displayprozessor, Bildspeicher, Grafikeinheit, Steuerrechner, Ein- und Ausgabegeräte, werden über den Systembus K 1600 bzw. UNIBUS/Q-Bus konfiguriert.
Der Displayprozessor ist ein Echtzeit-Pipelineprozessor, der die Bilder zeitlich synchron zur Videonorm verarbeitet. Die interaktive Verarbeitung erfolgt nach dem SIMD-Prinzip. Die Bildspeicher haben eine Kapazität bis zu 2 Megabyte.
Die Grafikeinheit realisiert typische Funktionen der Rastergrafik und der Grafiküberlagerung.
Der Steuerrechner hat eine Verarbeitungsbreite von 16 bit parallel und eine Hauptspeicherkapazität von 256 KByte.

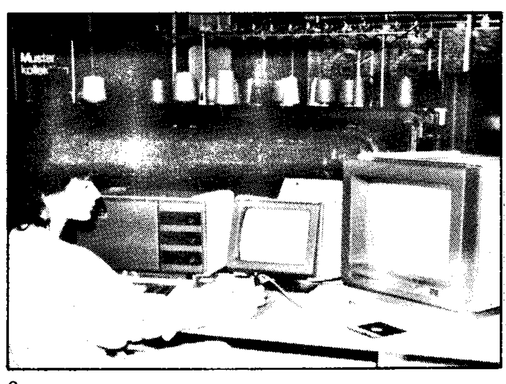
VEB Robotron-Vertrieb Berlin
Mohrenstraße 62
Berlin
DDR – 1080

Exporteur:
Robotron Export-Import
Volkseigener Außenhandelsbetrieb
der Deutschen Demokratischen Republik
Allee der Kosmonauten 24
Berlin
DDR – 1140

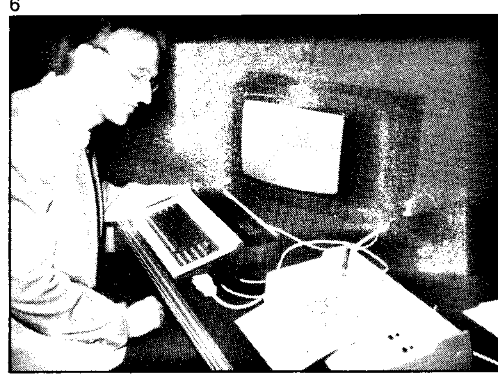
DEWAG LEIPZIG



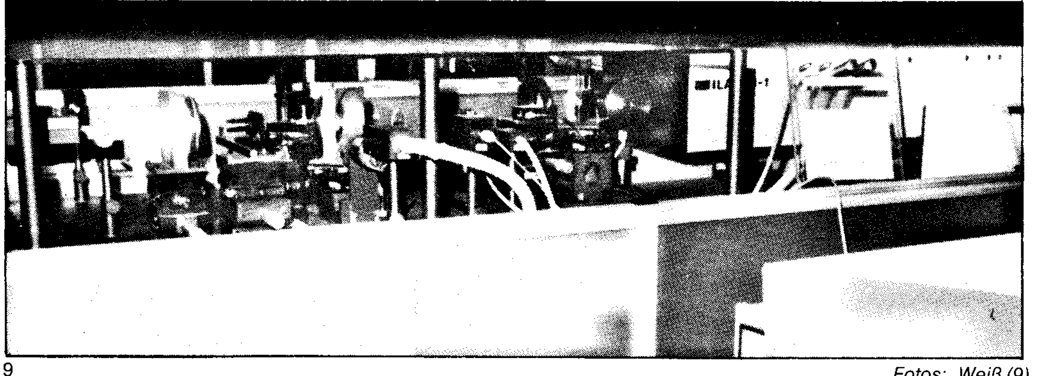

5


6


7


8

FÜR DIE LESER DER
MIKROPROZESSORTECHNIK
GEZEICHNET MIT
KLEINCOMPUTER KC 87
AUF
KLEINPLOTTER XY 4131
WÄHREND DER
AUSSTELLUNG
WISSENSCHAFT UND PRODUKTION
DER DDR IM DIENSTE DES VOLKES


9

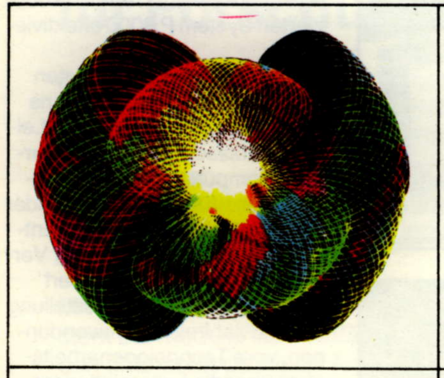
Fotos: *Weiß (9)*

(Fortsetzung von der 2. Umschlagseite)
Programmentwicklung und Test lassen sich künftig auch mit dem im Kombinat VEB Elektroapparatewerke Berlin-Treptow produzierten System P 8000 effektivieren.

Die Aufzählung der gezeigten Rechnertypen belegt, daß uns Computer unterschiedlicher Leistungsklassen – vom 8-Bit-Personalcomputer bis zum 32-Bit-Rechner – für die Steigerung der Effektivität von Forschung, Entwicklung und Produktion zur Verfügung stehen. Demonstriert wurde dieses auf der Ausstellung anhand zahlreicher Anwendungen, vom Technologenarbeitsplatz über die Leiterplattenentwicklung bis zu speziellen Aufgabenlösungen in der Forschung. Immer wieder beeindruckend sind die Beispiele, die unser tägliches Leben berühren, wie etwa die von einem K 1520 gesteuerte Lese- und Erkennungseinrichtung zum Briefverteilautomaten aus dem Institut für Post- und Fernmeldewesen (Bild 5). Mit ihr soll die Verteilleistung von bisher 3600 (halbautomatisch) auf 10 000 Sendungen je Stunde erhöht werden. Oder die Mustervorbereitungsanlage CAD-COMNIT, mit der ein wesentlich schnellerer und damit modisch aktueller Entwurf von Strickwaren am Computer möglich wird (Bild 6). Wie die Kleincomputer in der Ausbildung von Schülern und Lehrlingen genutzt werden können, wurde im Ausstellungsbereich Volksbildung gezeigt (Bild 7), kam aber auch bei der Betätigung der Jugendlichen in den Computerzentren zum Ausdruck. Hier waren einige der Kleincomputer übrigens mit dem tschechischen Plotter XY 4131 ausgestattet worden, der auch für den Vertrieb in der DDR vorgesehen ist (Bild 8).

Zum Abschluß unseres Berichtes von der Ausstellung ein kleiner Blick in die Zukunft und ein Beispiel für die Zusammenarbeit von Wissenschaft und Produktion: Im Komplex des Hoch- und Fachschulwesens gab es die experimentelle Demonstration eines neuen physikalischen Prinzips zu sehen, das den Ausgangspunkt für eine digitale Logik bildet – optische Bistabilität eines Typs, der 1982 an der Humboldt-Universität zu Berlin entdeckt wurde und an deren Erforschung in Kooperation mit dem Kombinat Mikroelektronik gearbeitet wird (Bild 9). Vielleicht zeichnet sich hier einer der Wege zum noch leistungsfähigeren optischen Computer ab, für deren Entwicklung in der ganzen Welt geforscht wird. *We*

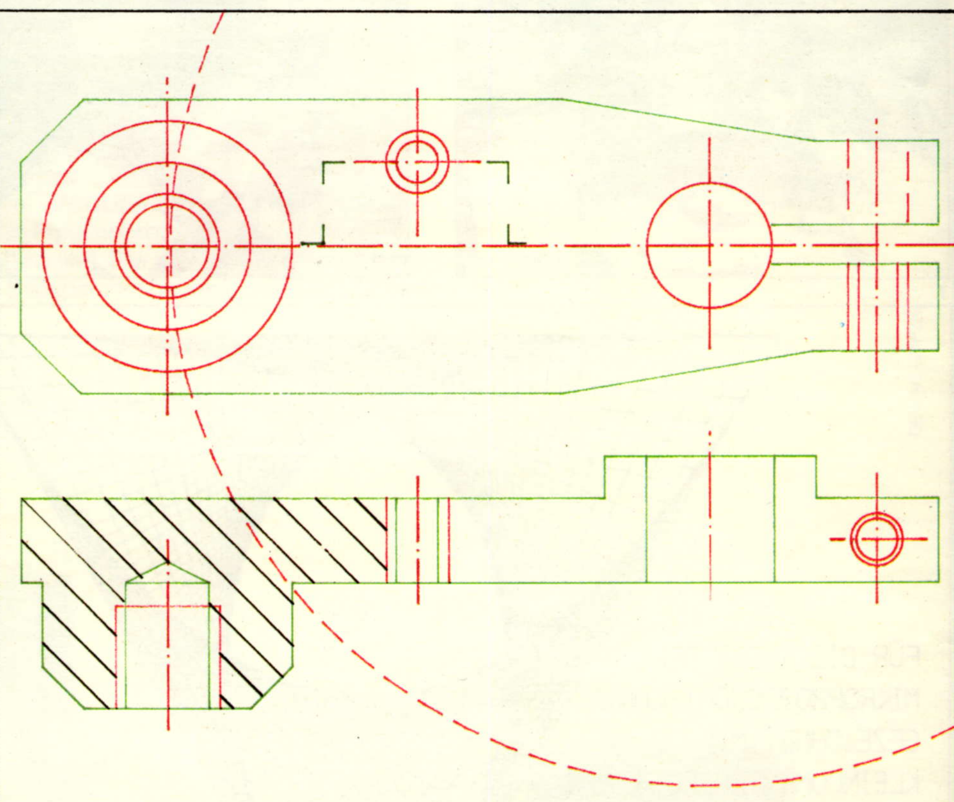
ISSN 0232-2892
Mikroprozessortechnik, Berlin 1 (1987) 9
S. 257–288

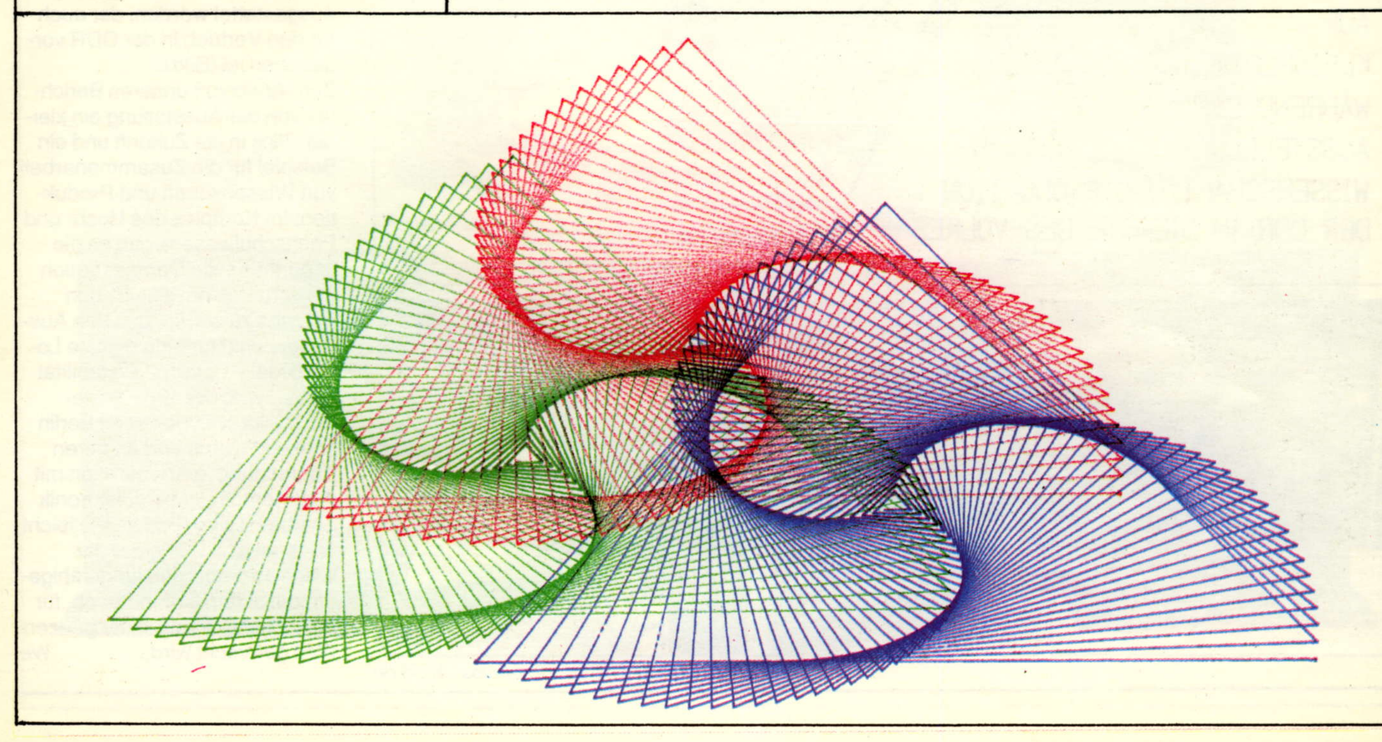


Ausstell

Wissenschaft und Produktion der DDR im Dienste des Volkes

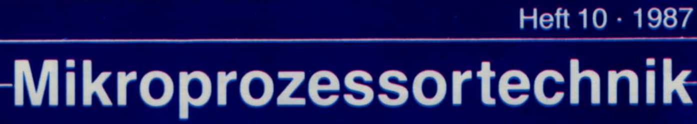
Ein Schwerpunkt angewandter Mikroelektronik in der Ausstellung – über die Sie weitere Informationen auf der 2. und 3. Umschlagseite dieses Heftes finden – waren CAD/CAM-Arbeitsstationen. Sichtbarer Ausdruck – im doppelten Sinne des Wortes – für das Vermögen dieser Anlagen, selbst vielschichtige Zusammenhänge anschaulich, aussagekräftig und eindeutig auszuweisen, waren die zahlreich erzeugten grafischen Darstellungen auf Druckern, Plottern und – siehe Titelbild – Laser-Beschriftungsanlagen. Die oftmals bestaunte ästhetische Attraktivität vieler Darstellungen, von denen wir Ihnen einige zeigen, sollte allerdings nicht die Aufgaben für Entwurf und Konstruktion vergessen lassen, deren Lösung ohne Computer heute meist unmöglich wäre.





ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP



Heft 10 · 1987

Mikroprozessortechnik

VEB Verlag Technik Berlin
ISSN 0232-2892

Schaltkreisentwurf

- Modulgenerierung auf Leafcellbasis
- Schaltungsentwurf im Top-Down-Stil




Info

Neuer sowjetischer Computer

Kurz vor dem Abschluß stehen im Moskauer Institut für Feinmechanik und Rechentechnik Arbeiten an einem neuen Computer, der mehrere Milliarden Rechenoperationen in der Sekunde ausführen kann. Bei Elbrus-3 handelt es sich nach TASS um einen Computer einer neuen Generation, dessen Schnelligkeit aufgrund des Modulprinzips seines Aufbaus gewährleistet wird. Er enthält 16 Prozessoren, die zusammengeschlossen sind und bei einem auftretenden Defekt jeweils die Aufgabenlösung eines anderen Prozessors mit übernehmen können. Daraus resultiert neben der hohen Leistung seine große Betriebssicherheit. Die modernsten sowjetischen Computer enthielten bisher 10 Prozessoren und konnten 125 Millionen Operationen in der Sekunde ausführen. Mit der Anwendung des Modulsystems gehen die sowjetischen Wissenschaftler eigene Wege bei der Entwicklung von Computern hoher Leistung. Der aus mehreren Prozessoren bestehende Computeraufbau diene dem Ziel, Computer zur Parallelbearbeitung von komplizierten Algorithmen und mathematischen Prognosen zu entwickeln.
Das sowjetische Mehrprozessor-Prinzip habe bei amerikanischen Fachleuten Interesse hervorgerufen, die zur Zeit am Cray-3 mit 16 Prozessoren arbeiten.

ADN



Das Gesicht der neuen bulgarischen Zeitschrift „Computer für Euch", über die wir in MP 9/87, S. 286 informierten. In populärwissenschaftlicher Herangehensweise werden in dieser 32 Seiten umfassenden Publikation Hardware- und Softwarethemen behandelt.

Repro: Wosnizok

BASIC – 1×1 des Programmierens

In einer Koproduktion von Radio DDR II und dem VEB Deutsche Schallplatten wird auf 6 Kassetten ein Lehrgang zum Erlernen von BASIC – auf der Grundlage des BASIC-Kurses von Prof. Dr. Horst Völz im Rundfunk – herausgegeben. Die Kassetten 1 bis 5 enthalten Erläuterungen zu den BASIC-Anweisungen und Funktionen. Hinweise für eine effiziente Programmgestaltung findet der Hörer auf Kassette 5. Die gesendeten Programme enthält Kassette 6. Die Programmbeispiele liegen bisher für KC85/2 und /3 bzw. KC85/1 und 87 vor. In Vorbereitung befinden sich jeweils Kassetten mit den Programmen lauffähig auf Commodore C 64, Atari 800XL und Sinclair Spectrum.
Bezogen werden kann die Präsentbox (6 Kassetten und 2 Texthefte) zu einem Preis von 96,60 M über den **Schallplatteneinzelhandel** (Katalognummer: 003002-007).

Hamann

Ausstellung über Bürotechnik

Vom 17. bis 19. November 1987 zeigt die Fa. RANK XEROX in Berlin in einer Ausstellung ihre neuen Produkte.
Neben den Kopierern 1065 und 1012 sowie dem Telekopierer 7010 werden vorgestellt:

– IBM-AT-kompatibler 16-Bit-PC OPat, mit Matrixdrucker und Software, insbesondere Desktop-Publishing-Programm: Ventura Publisher
– MS-Net-kompatibles Hi-Net-PC-Local-Area-Network
– 4045-Laserdrucker und -kopierer
– CAD-Arbeitsplatz, bestehend aus dem PC OPat mit 19"-Grafik-Bildschirm, AUTOCAD-Software, Grafiktablett mit Lightpen und Penplotter; außerdem elektrostatische Plotter von VERSATEC und RX-Plandrucker.

In allen drei Tagen stehen Systemspezialisten zur Verfügung.
Interessierte Betriebe können sich zwecks Teilnahme bzw. Einladung an das RANK-XEROX-Büro im Internationalen Handelszentrum in 1086, Berlin, Friedrichstr., Tel: 20 96 26 75/76 wenden.

M. Mittelhaus

Aus der Arbeit der WGMA

Anläßlich der Aktivtagung der WGMA am 18. Juni 1987 in Leipzig wurde die Bildung neuer zukunftsträchtiger Fachgremien beschlossen, und es wurden deren Vorsitzende berufen:

FA Beratungs-/Expertensysteme in der Automatisierungstechnik
– unter der Leitung von Prof. Dr. sc. Jürgen Wernstedt
FA Automatisierungssysteme
– unter der Leitung von Prof. Dr. sc. Peter Neumann
FUA Basissteuerungen
– unter der Leitung von Dr. sc. Jan Lunze
FUA Betrieb von Kleinrechnern
– unter der Leitung von Dr. Tim Retter
FUA Modellierung und Simulation in der Ökonomie
– unter der Leitung von Prof. Dr. sc. J. A. Müller.

Mit der Bildung dieser Fachgremien wird das Ziel verfolgt, wissenschaftlich-technische Entwicklungen frühzeitig in die Arbeit der WGMA zu integrieren. Die sich hier entwickelnde interdisziplinäre Gemeinschaftsarbeit auf diesen Fachgebieten soll mit dazu beitragen, die sozialistische Zusammenarbeit zu entwickeln, um mit KDT-Standpunkten und -Empfehlungen staats- und wirtschaftsleitende Einrichtungen bei der Entscheidungsfindung zu unterstützen.
Ebenso gilt es, mit diesen Gremien den notwendigen Bildungsvorlauf für die Weiterbildung in der KDT zu schaffen als eine notwendige Konsequenz der weiteren Forcierung von Wissenschaft und Technik.

K.-D. Müller

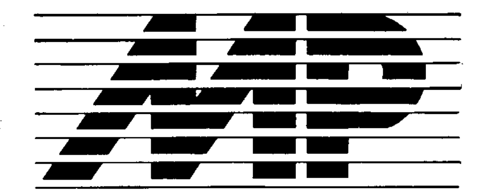
Industriestandard für optische Speicher?

Die wichtigsten amerikanischen und japanischen Hersteller haben sich auf ein gemeinsames Aufzeichnungsformat bei 5,25-Zoll-Laufwerken geeinigt, so die Zeitschrift Elektronik in ihrer Ausgabe 14/87. Damit wird in Zukunft ein Austausch von Disketten zwischen Laufwerken verschiedener Fabrikate möglich – sowohl für einmal beschreibbare wie für löschbare und ausschließlich lesbare optische Datenträger. Das dem Komitee X3B11 des American Accredited Standards Committee vorgeschlagene Composite/Continuous-Format bietet im Vergleich zu anderen Verfahren einige Vorzüge, wie etwa höhere Speicherkapazität und Übertragungsraten oder kürzere Zugriffszeiten.
18 Hersteller haben sich auf das Composite/Continuous-Format geeinigt. Grundlage des Composite/Continuous-Verfahrens ist das fortlaufende Nachführen und Scharfstellen des Kopfes. Es erlaubt die Abspeicherung von mindestens 300 MByte auf jeder Seite einer 5,25-Zoll-Diskette. Das stetige Taktsignal schafft die Voraussetzung für eine hohe Übertragungsrate. Zeitliche Abweichungen zwischen dem Datentakt und dem vorformatierten Takt, wie sie beim Sampled-Servo-Format auftreten können, sind damit ausgeschlossen. Da die fortlaufenden Rillen außerdem ein schnelleres Überspringen der Spuren gestatten, ergeben sich kürzere Zugriffszeiten.

Drucker schreibt mit fester Tinte

Die US-amerikanische Firma dataproducts corp. aus Woodland Hills/Kalifornien hat einen Drucker entwickelt, der aufgrund einer neuartigen Drucktechnik mit fester Tinte Briefqualität erreicht, meldete die Zeitschrift Electronics. Die Nachteile herkömmlicher Tintenstrahldrucker – wie verstopfte Düsen und die Verwendung nur bestimmter Papiersorten – sollen dabei aber vermieden werden. Der neue Drucker verwendet eine feste Tinte in Pelletform und einen neuentwickelten Druckkopf, der in Briefqualität eine Geschwindigkeit bis 400 Zeichen/s ermöglicht. Die horizontale Auflösung beträgt bei 200 Zeichen/s 480 Punkte/Zoll und bei 400 Zeichen/s 240 Punkte/Zoll. Die vertikale Auflösung liegt bei 240 Punkte/Zoll. Insgesamt trägt der Druckkopf 32 Düsen. Die Druckqualität soll besser als bei Typenrad- oder Laserdruckern sein, und die Geräuschentwicklung wird mit weniger als 55 db angegeben. Ein weiterer Vorteil ist, daß ein breites Sortiment an Papier und auch transparente Vorlagen bedruckt werden können. Bei dem eigentlichen Druckvorgang werden die Tintenpellets geschmolzen und verflüssigt auf das Papier gespritzt, wo sie zu feinen, kräftig dunklen Punkten erstarren.

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0232-2892



Mikroprozessortechnik, Heft 10 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR-1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 2 87 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)
Titelfoto: Werkfoto

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

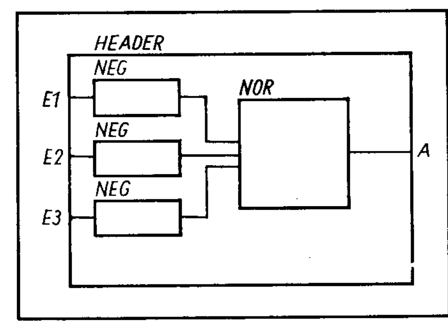
Redaktionsschluß: 25. August 1987

AN (EDV) 49837

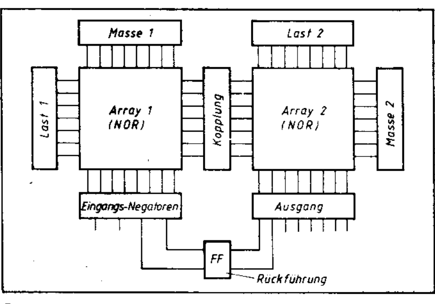
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

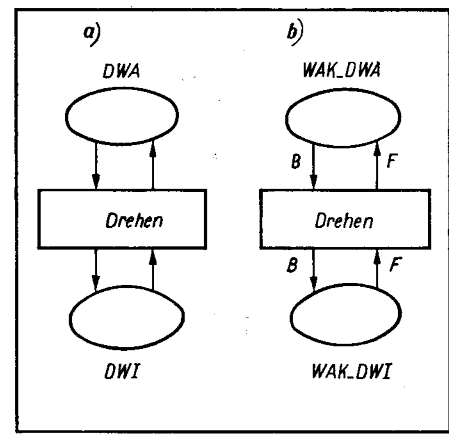
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tisku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hà Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR-7010, und Leipzig Book Service, Talstraße 29, DDR-7010 Leipzig



Seite 291



Seite 293



Seite 299

Vorschau

In MP 11/87 haben wir für Sie einen Schwerpunkt zum Thema Computergrafik vorbereitet; u. a. finden Sie folgende Beiträge zu dieser Thematik:
– Pseudografik auf PC 1715 und A 7100
– Grafik am A 7100
– Grafikprogramm zur Darstellung von Ergebnissen der Aufwärtsübersetzung.
Weiterhin informieren wir Sie mit einem Einführungsbeitrag über die Programmiersprache PROLOG und drucken einen Disassembler für den KC 85/3 ab.

Inhalt

✉ Zu RAM-Floppy – ein schneller Zusatzspeicher für Bürocomputer in MP 3/87, Seite 83

Zu diesem Aufsatz, dessen Anliegen uneingeschränkt zuzustimmen ist, gibt es einige Bemerkungen bezüglich der hard- und softwaretechnischen Realisierung.
Der Aufbau einer RAM-Diskette mit speziellen Steckeinheiten (dyn. Seitensteuerung) ist sicherlich nicht von der Mehrzahl der Anwender zu akzeptieren. Vielmehr sollte es gelingen, solche Steckeinheiten dafür einzusetzen, die zur Zeit von industriellen Herstellern (z. B. Erweiterungsmodul EM 256, Kombinat Robotron) bzw. Zentren für wissenschaftlichen Gerätebau (z. B. 256 KByte NANOS, IHS Warnemünde/Wustrow) hergestellt werden. Dabei ist dann das Prinzip der Zwischenpufferung der Daten im nicht geschalteten Systembereich anzuwenden (siehe auch /1/). Als Zwischenpuffer können z. B. in CP/M-Systemen die Speicherbereiche für die Read-after-Write-Funktion der Diskettentreiber genutzt werden. Die Verwendung der Blockpuffer der Diskettentreiber wird nicht empfohlen, da dieses bei ungeschickter Anwenderprogrammierung ein ständiges Leeren der Blockpuffer verursachen würde.
Es ist damit zu rechnen, daß sich die eigentliche Blocktransportzeit gegenüber der Anwendung einer dynamischen Seitensteuerung verdoppelt. Da jedoch für die Berechnung der Speicheradressen und der Prüfsummen sowie die Datenverifizierung durch Read-after-Write ebenfalls Rechenzeit benötigt wird, wirkt dieser Nachteil nicht so stark.
Beim Zugriff auf die RAM-Diskette ist zu beachten, daß im wegzuschaltenden Hauptspeicherbereich Interruptroutinen der Anwenderprogramme liegen können. Für die kurze Zeit des Datentransportes zwischen der RAM-Diskette und dem Puffer ist der Prozessor für Interrupts zu sperren.

Dr. Karl Westendorff, Wismar

Literatur

/1/ Westendorff, Karl: COS/PSA und COS/MFA – Zwei neue Betriebssysteme für das Meßcomputersystem PSA und das modulare Fourieranalysesystem MFA. Tagungsbeiträge der Fachtagung „Computer und Mikroprozessortechnik '86", KDT, Fachverband Elektrotechnik

✉ CP/M oder UDOS?

Im Beitrag *Nachladbare Gerätetreiber für Personal- und Bürocomputer* in MP 6/87 wird geschildert, wieviel Überlegungen, Softwareentwicklungen und Tricks erforderlich sind, um die Einbindung zusätzlicher Gerätetreiber in CP/M-kompatible Betriebssysteme zu ermöglichen.
In dem Beitrag sind sieben allgemeingültige Forderungen an nachladbare Gerätetreiber herausgearbeitet. Die geforderten Leistungsmerkmale sind notwendig, werden aber von CP/M und kompatiblen Betriebssystemen in keiner Weise unterstützt.
Es ist verwunderlich, wie CP/M mit den genannten und weiteren schwerwiegenden konzeptionellen Schwächen überhaupt so weite Verbreitung finden konnte und noch heute genutzt wird; erklärbar nur dadurch, daß CP/M schon auf dem inzwischen veralteten I8080/8085 Prozessor lauffähig war. Das RIO-kompatible Betriebssystem UDOS erfüllt alle im Beitrag genannten Forderungen:
① Der Hauptspeicherbereich ist in 512 Seiten zu je 128 Byte aufgeteilt, die vom Speicherverwalter einzeln oder als zusammenhängende Segmente verwaltet werden. Von der Anforderung bis zur Freigabe reserviert der Speicherverwalter RAM-Segmente exklusiv für das anfordernde Programm.
② Vom Betriebssystem, Systemprogrammen und Treiberroutinen benötigte Speicherbereiche werden ebenfalls vom Speicherverwalter reserviert; Konflikte sind ausgeschlossen.
③ Massenspeicherzugriff und Ein-/Ausgabe werden als einheitliche Operationen aufgefaßt! (Ist z. B. das Ausstanzen eines Lochstreifens, Archivieren und Wiedereinlesen eine Speicherung oder Ein-/Ausgabe?). Gerätename und Laufwerksnummer sind Bestandteil des Filenamens. Damit kann der Bediener beliebige Speicher- bzw. E/A-Geräte anwählen.
④ Laden neuer Treiberprogramme mit Speicherreservierung und Einbinden ins Betriebssystem mittels Systemkommando ACTIVATE.
⑤ DEACTIVATE entfernt die Treiber wieder.
⑥ Anzeige aller aktiven Geräte durch das Systemkommando LADT.
UDOS hat weitere Vorzüge, wie z. B. für jede Datei freie Wählbarkeit des Disketten-Aufzeichnungsformats zwischen 128 Byte und 4 KByte je Block zur Optimierung zwischen guter Speicherausnutzung und schnellem Zugriff, Schutzeigenschaften für jede Datei (löschgeschützt, schreibgeschützt...), Kommandosprache zur Formulierung häufig gebrauchter Abläufe wie Compilieren/Assemblieren/Binden und anschließendes selbsttätiges Abarbeiten ohne zwischenzeitlichen Bedienereingriff u. v. a. m.
Das UDOS-Betriebssystem ist auf MRES, BC 5120, P 8000 und PC 1715 implementiert.

Archibald Hoklas, Rostock

Wir teilen Ihre Auffassungen bezüglich der besseren technischen Leistungsmerkmale des Betriebssystems UDOS. Es ist aber sicher kein Geheimnis, daß sich international für 8-Bit-Mikroprozessorsysteme nicht UDOS oder RIO, sondern CP/M in überwältigender Weise durchgesetzt hat. Man mag das beklagen, unabhängig davon muß man aber dieser Entwicklung auch in der DDR durch Einsatz des CP/M-kompatiblen SCP Rechnung tragen. Die Vor- und Nachteile eines Betriebssystemkonzeptes für den Anwender lassen sich eben nicht nur rein technisch vom Betriebssystemkern her beurteilen. Natürlich bleibt letztlich jedem Anwender die Qual der Wahl nicht erspart.



Am 7. August 1987 hat sich das Leben unseres Freundes und Genossen Manfred Schubert, Präsident der Kammer der Technik, Abgeordneter der Volkskammer der DDR, Ordentlicher Professor an der Technischen Universität Dresden, Korrespondierendes Mitglied der Akademie der Wissenschaften der DDR, Ordentliches Mitglied der Sächsischen Akademie der Wissenschaften, nach schwerer Krankheit im Alter von 57 Jahren vollendet.
Wir verneigen uns in tiefer Dankbarkeit vor seiner Persönlichkeit. Mit Manfred Schubert verlieren wir einen national und international geschätzten Wissenschaftler unseres Landes, der seine ganze Persönlichkeit für unser sozialistisches Vaterland, für den Sozialismus und den Frieden einsetzte.
Manfred Schubert hat als Präsident der Kammer der Technik in den vergangenen 13 Jahren einen hohen persönlichen Beitrag zur Entwicklung unserer sozialistischen Ingenieurorganisation geleistet.
Vor Ort in den Kollektiven, bei den Menschen am Arbeitsplatz, das war sein Wirkungsfeld. Ob unter Tage im Kaliwerk Zielitz, im Kombinat Elektro-Apparate-Werke „Friedrich Ebert" Berlin, im VEB Geräte- und Regler-Werke Teltow oder in der Betriebssektion der KDT im Chemiefaserkombinat „Wilhelm Pieck" Schwarza, überall und immer war sein Rat gefragt.
Der sparsame und behutsame Umgang mit den begrenzten Reichtümern der Natur und die damit verbundene Verantwortung für die Generationen nach uns waren ihm ein hohes persönliches Anliegen. Darin sah er eine große gesellschaftliche Verantwortung des Ingenieurs.
Mit seinem überzeugenden politischen Engagement, seinem unermüdlichen Fleiß und seinem hohen Verantwortungsbewußtsein für die Entwicklung von Wissenschaft und Technik, mit seiner menschlich so aufgeschlossenen Art, seinen anregenden, auf die Lösung der Probleme gerichteten Hinweisen, zog Manfred Schubert jeden in seinen Bann und motivierte ihn zu neuen Leistungen. Jede Aufgabe betrachtete er als seinen persönlichen Auftrag und setzte sich leidenschaftlich für die Erfüllung ein.
Mit uns trauern Genossen und Freunde, Mitglieder der Kammer der Technik, Wissenschaftler, Ingenieure und Studenten an den Forschungs- und Bildungseinrichtungen sowie in den Kombinaten und Betrieben unseres Landes.
Wir erfüllen sein Vermächtnis, wenn wir in seinem Sinne die vor uns stehenden Aufgaben lösen, und gedenken seiner in Dankbarkeit und Ehrerbietung.

Präsidium der Kammer der Technik

Programmsystem für den Schaltkreisentwurf im Top-Down-Stil

Matthias Baier, Michael Elgner
Zentralinstitut für Kybernetik und Informationsprozesse der AdW der DDR, IT Dresden

1. Entwurfsschritte

Ausgangspunkt des hier vorgestellten Schaltkreis-Entwurfs ist eine Schaltungsbeschreibung auf höherem Abstraktionsniveau, z. B. ein Logik-Schaltplan bzw. ein Elektrik-Schaltplan. Ein solcher Plan ist dadurch gekennzeichnet, daß er aus logischen oder elektrischen Elementen und der Angabe der Verbindungen zwischen diesen besteht. Dabei ist die Anordnung der Elemente auf dem Plan im allgemeinen so, daß ein Ingenieur leicht das funktionelle Zusammenspiel der Schaltplanelemente ableiten kann (ingenieurnahe Darstellung). Solche Schaltplanelemente können sein:

- Symbole, die wiederum für einen Schaltplan stehen, wodurch eine hierarchische Beschreibung komplexer Schaltungen durch weniger Schaltungsteile möglich ist und/ oder
- Primitives, also Funktionseinheiten, die selbstdefinierend und nicht weiter strukturell spezifiziert sind. Unter solche Primitives, die nichts anderes sind als elementare Beschreibungen, fällt auch der Transistor.

Da ein Schaltplan selbst aus Schaltplänen bestehen kann, ist es möglich, immer komplexere Funktionseinheiten zu beschreiben. Dabei entsteht eine Hierarchie von Schaltplänen, die in ihrer Gesamtheit einen komplexen Sachverhalt detailliert darstellt (Bild 1).
Die hier behandelte Entwurfsstrategie beinhaltet nun die computergestützte Überführung einer solchen Hierarchie gegebener Schaltpläne von einem hohen Abstraktionsniveau bis in das Feinlayout. Dies geschieht in zwei Richtungen:
– durch Spezifizierung der funktionellen Struktur, indem zu jedem Schaltplanelement, falls noch nicht vorhanden, ein neuer Schaltplan angegeben wird, womit eine Schaltplanhierarchie entsteht
– durch Übergang von einem höheren zu einem niedrigeren Abstraktionsniveau. Dies geschieht auf interaktiv-graphischem Wege durch schrittweise Spezifizierung der Schaltplanelemente, also der Zuführung von Informationen, die für die immer niedrigeren Abstraktionsebenen notwendig sind.
Der Entwurf von einem Logik-Schaltplan aus erfolgt schrittweise, wobei die Einzelschritte nachfolgend charakterisiert werden.

1.1. Logik-Plan-Protection

Ist ein Logikplan (z. B. durch Simulation) für richtig befunden, so erhält er einen Schutzcode, der die relevante logische Struktur dagegen schützt, daß sie im weiteren Entwurf durch Entwerferhandlungen verändert wird.

1.2. Überführung des Logik-Schaltbildes in den Logik-Floor-Plan

Ist ein Schaltplan dadurch gekennzeichnet, daß er bestimmte Informationen so darstellt, daß ein Ingenieur leicht den logischen Informationsfluß erfassen kann (das heißt, die Schaltplanelemente sind dem Signalfluß entsprechend angeordnet), so ist der Floor-Plan dadurch charakterisiert, daß nunmehr die prinzipielle Anordnung der Schaltungselemente so dargestellt wird, wie sie später auf dem Chiplayout auch sein wird. Man geht mit dem Floorplan damit von der ingenieurorientierten Darstellung über zur chiporientierten Darstellung (Bild 2). Der Floor-Plan ist gekennzeichnet durch
– chipnahe Größe der verwendeten Funktionsblöcke sowie der Anordnung der Ausgänge/Eingänge inclusive des übergeordneten (diese Funktionsblöcke zusammenfassenden) Blocks.
– chipnahe Anordnung der Funktionsblöcke im übergeordneten Block, wobei aber in der Regel noch keine chipnahe Darstellung der Realisierung der Signalverläufe, sondern immer noch Darstellung des logischen Signalflusses in ingenieurnaher Weise erfolgt.

1.3. Überführung des Logik-Floor-Planes in den Elektrik-Floor-Plan

In diesem Entwurfsschritt wird der Logik-Floor-Plan um die für die elektrische Funktion unerläßlichen Potentiale (Masse, Betriebsspannung) erweitert, indem neue, nur auf dem Elektrik-Niveau relevante Anschlüsse an den Symbolblöcken definiert und miteinander verbunden werden (Bild 3).
Analog zur Logik-Plan-Protection wird ein (nach erfolgreicher Simulation als korrekt bestätigter) Elektrik-Plan gegen Veränderungen im weiteren Entwurfsablauf geschützt.

1.4. Übergang in das Layout-Niveau

Das Layout-Niveau ist durch folgende Fakten gekennzeichnet:

- Unmöglichkeit der Veränderung des logischen und elektrischen Netzes
- Vorgabe von Anweisungen für die Realisierung der Verdrahtung auf dem Chip, insbesondere Vergabe der technologischen Ebene und der Lage der Anschlüsse des umschließenden Funktionsblockes (Header)
- Während im Sinne des elektrisch/logischen Netzes Signal- oder Potentialein-/Ausgänge nicht mehr verändert werden, können doch zu diesen Ein-/Ausgängen äquivalente definiert werden, die layoutbezogen andere Positionen bei gleicher elektrischer/logischer Bedeutung haben.
- Möglichkeit einer Umplazierung der Funktionsblöcke.

Aus diesen Informationen und dem elektrischen/logischen Netz wird dann automatisch ein kompaktes Feinlayout erzeugt.

Dies geschieht durch
– automatische Bestimmung des topologischen Leitbahnverlaufs bzw. Übernahme der expliziten Blockverbindung (s. 1.4.1.) entsprechend der angegebenen Potential- oder Signalverläufe
– Überführung des topologischen Leitbahnverlaufes in ein symbolisches Feinlayout (STICKS)
– Compaktion und Feinlayoutbestimmung mittels der Routinen der eingebundenen STICK-Software.

Durch die variable Angabe von Entwurfsregeln in der STICK-Software ist damit ein weitgehend technologieunabhängiger Entwurf (bei Beibehaltung der Basis-Technologie) möglich.
Für die oben angeführte interaktive Beeinflussung der Verdrahtung unterscheidet man die folgenden zwei Arbeitsweisen.

1.4.1. Explizite Blockverbindung

Die explizite Blockverbindung (Bild 4) ist gekennzeichnet durch
– konkrete Angabe des Verlaufs der Leitbahnen gemäß der topologischen Lage der Funktionsblöcke
– konkrete Vorgabe, in welcher technologischen Layoutebene die betrachtete Leitbahn realisiert werden wird.
Explizite Blockverbindungen haben den Vorteil, daß die Entwerfererfahrungen bei der Plazierung und Trassierung voll zur Geltung kommen können, solange die Komplexität gering bleibt. Dabei wird vom Programmsystem überwacht, daß die interaktiv eingegebene Leitbahnführung mit dem bereits abgespeicherten elektrischen Netz widerspruchsfrei bleibt.

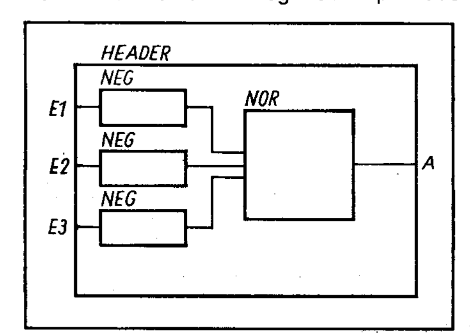


Bild 1 Ingenieurorientiertes Blockschaltbild

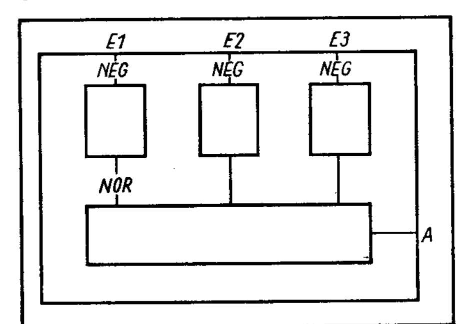


Bild 2 Chiporientierter Logik-Floorplan

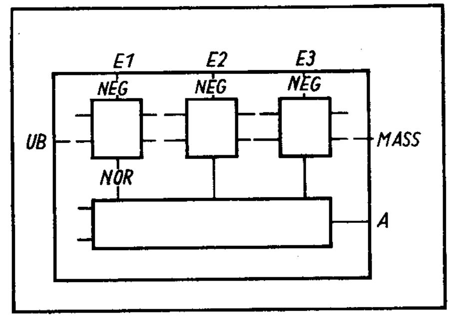


Bild 3 Chiporientierter Elektrik-Floorplan

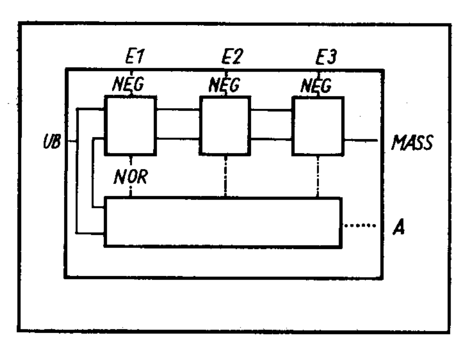


Bild 4 Symbollayout / explizite Blockverbindung

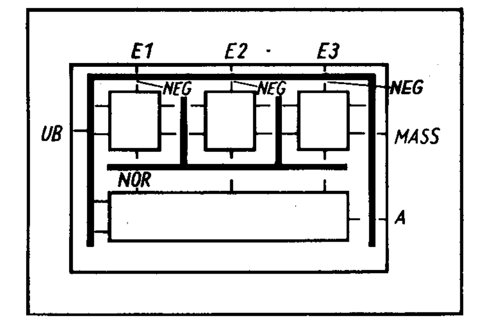


Bild 5 Symbollayout / implizite Blockverbindung

1.4.2. Implizite Blockverbindung durch Angabe von Verdrahtungskanälen

Diese Methode ist gekennzeichnet durch Angabe des Raumes zwischen den Funktionsblöcken, der für die Verdrahtung benutzt werden soll (Mittellinien der Kanäle). Ein automatischer Prozeß ermittelt aus diesen Kanalangaben und dem elektrischen Netz den Leitbahnverlauf. Das geschieht in mehreren Etappen. Ein Globalroutingprozeß legt die Verdrahtungswege für jeden einzelnen Blockanschluß fest. Nach einer automatischen Prüfung auf ausreichenden Verdrahtungsraum und einer eventuell notwendigen Streckung auf symbolischem Niveau schließt sich das lokale Routing an, welches die optimierte Anordnung der Leitbahnen sowie deren technologische Ebenenzuordnung festlegt und ein entsprechendes STICK-Symbollayout erzeugt, welches anschließend kompaktiert und in ein entwurfsregelgerechtes Feinlayout überführt wird.

Die implizite Blockverdrahtung hat den Vorteil, daß sie dem Entwerfer den Arbeitsaufwand zum Legen der Verbindungen zwischen den Blöcken abnimmt. Dieser Vorteil ist insbesondere dann gravierend, wenn das Verbindungsnetzwerk aus Entwurfsschritten des Logik- und Elektrikniveaus bereits bekannt ist und nicht mehr interaktiv eingegeben werden muß (Bild 5).

1.5. Dynamische Bestätigungssimulation

Das Feinlayout liegt nun in den konkreten geometrischen Abmessungen vor. Damit kann es exakt automatisch vermessen werden. Da die Bezugnahme zwischen dem Feinlayout und den höheren Abstraktionsniveaus gewährleistet ist, ist es möglich, das logische und das elektrische Netz mit den dynamischen Parametern zu ergänzen und einer dynamischen Netzwerkanalyse zuzuführen.

2. Besondere Bedeutung dieses Top-Down-Entwurfes

Im hier beschriebenen Entwurf ist der Top-Down-Stil konsequent bis ins Maskenlayout geführt. Das bedeutet, daß mit Sicherheit ein einmal als korrekt befundener logischer oder elektrischer Schaltplan auch nach der Layout-Manipulation genau diesen Schaltplan im Chip abbildet. Damit zeichnet diesen Entwurf eine Reihe von *Besonderheiten* aus:

- Der Entwurf ist ein computergestützter interaktiv-graphischer Top-Down-Block-Entwurf.
- lückenloser Übergang von einem hohen abstrakten Entwurfslevel bis ins Maskenfeinlayout
- Der Entwurf ist verifikationsarm. Eine Verifikation ist nur dort nötig, wo eine neue Information dem Entwurf zugeführt wird.
- Angabe von Feinlayoutdaten auf symbolischem Niveau
- automatische Feinlayout-Generierung komplexer Funktionsblöcke
- Unterstützung der Bestätigungssimulation durch automatisches Vermessen des Feinlayouts und Ergänzung der strukturellen Netzbeschreibung mit automatisch entnehmbaren Parametern für die dynamische Logiksimulation (bzw. Netzwerkanalyse).

Der Nachteil eines solchen Entwurfs ist ein etwas erhöhter Layoutflächen-Bedarf, der aber allen automatischen Verfahren eigen ist. In dem hier vorgestellten System tritt dieser Verlust vor allem an den Rändern der Funktionsblöcke auf, also an den Stellen des Hierarchie-Interfaces, da dort zur Vermeidung von Layout-Entwurfsregel-Fehlern notwendige Schutzzonen beachtet werden müssen.

KONTAKT

Akademie der DDR,
Zentralinstitut für Kybernetik und Informationsprozesse, Institutsteil Dresden,
Haeckelstr. 20, Dresden, 8027;
Tel. 4633164

Filetransfer zwischen PC/BC und ESER-Rechnern

Grundlage für diesen Filetransfer ist der EC 7925-Emulator des VEB Chemieanlagenbaukombinat Grimma. Dieser Emulator realisiert die Übertragungsprozedur BSC3 unter dem Betriebssystem SCP. Der PC 1715 bzw. BC A5120/A5130 ist damit als fernangeschlossenes Terminal (EC 7921) mit 1xEC7927 für OS(TSO), SVM(PTS) und DOS-3 einsetzbar. Beschrieben wird der PC bzw. BC im MCP für TSO bzw. im DMKRIO nicht als EC 7925, sondern als EC7921 mit 1xEC7927, da er über einen Ltg.-Puffer von 1920 Byte verfügt. Dieser Terminal-Emulator enthält zusätzlich eine Filetransfer-Schnittstelle, die ESER-seitig unter DOS-3 mit MOVESTAR/DOS-3 des VEB Chemieanlagenbaukombinat Grimma unterstützt wird.

Die Unterstützung der gleichen Schnittstelle – ab Version 5 des Emulators – im OS(TSO) und SVM(PTS) erfolgt durch MOVESTAR/OS bzw. MOVESTAR/SVM. Beide Programme wurden vom Karl-Weierstraß-Institut für Mathematik der AdW in Zusammenarbeit mit dem VEB Chemieanlagenbaukombinat Grimma entwickelt.

1. OS-Version unter TSO

- Aufruf des Filetransfers als TSO-Kommando MOVESTAR
- Mögliche Funktionen sind:
  - Anzeigen Diskettenverzeichnis
  - Löschen Diskettendatei
  - Übertragen Textdatei PC (BC) ↔ ESER
  - Übertragen Binärdatei PC (BC) ↔ ESER
- Textdateien auf PC (BC) sind Dateien, die druckbare Zeichen (20H ... 7FH im KOI-7-CODE) enthalten. Ebenfalls werden Steuerzeichen (00H ... 1FH) außer 03H und 17H übertragen (1AH darf nur letztes Zeichen der Datei sein). Bei Zeichen von 80H ... FFH wird das höchstwertige Bit gelöscht. Als Satzendekennzeichen gilt die Kombination CR/LF (0D0AH). Steuerzeichen (00H ... 1FH) werden nicht in die OS-Datei übernommen.
- Binärdateien können alle Zeichen 00H ... FFH enthalten.
- Dateiattribute unter OS:
  - Als Satzformate sind F, FB, V, VB, U möglich.
  - Die OS-Datei muß angelegt und katalogisiert sein.
  - Die logische Satzlänge richtet sich nach den DCB-Parametern der OS-Datei.

Bei Textdateien bedeutet das bei fester Satzlänge:
Ist der PC(BC)-Datensatz kürzer, wird der OS-Datensatz mit Blanks aufgefüllt.
Ist der PC(BC)-Datensatz länger, werden mehrere OS-Datensätze erzeugt.

- Alle für die Übertragung notwendigen Angaben werden im Dialog vom Nutzer abgefragt.

2. SVM-Version unter PTS

- Analoger Funktionsumfang wie im OS.
- Das erzeugte PTS-File hat das Satzformat V. Eine Umformung in andere Satzformate ist unter PTS unkompliziert.
- Ist das PTS-File das Quellfile, so sind die Formate F und V zugelassen.

3. BTAM-Version unter OS

Eine TSO-unabhängige Version für OS unter Nutzung der Zugriffsmethode BTAM befindet sich in Vorbereitung.

4. Nachnutzung

- Vertreiber des EC 7925-EMULATORS und MOVESTAR/DOS-3 ist der VEB Chemieanlagenbaukombinat, Stammbetrieb, ORZ, Bahnhofstraße 3–5, Grimma, 7240; Tel. 632334 (Koll. Hinz)
- Vertreiber des Filetransfers MOVESTAR/OS und MOVESTAR/SVM ist die Akademie der Wissenschaften der DDR, Karl-Weierstraß-Institut für Mathematik, Rechenzentrum, Mohrenstraße 39, Berlin, 1086; Tel. 2077586 (Koll. Vetter)

H. Vetter

Modulgenerierung auf Leafcell-basis

Dr. Frank Lenke
Zentralinstitut für Kybernetik
und Informationsprozesse der AdW der DDR, Institutsteil Dresden

1. Einordnung der Modulgenerierung in den Problemkreis LSI-/VLSI-Entwurf

Im Zuge der LSI- und VLSI-Entwicklung entstanden und entstehen zahlreiche hochgradig automatisierte Entwurfssysteme. Dazu zählen auf jeden Fall auch Siliconcompiler und -assembler /1/. Die Ansprüche, die derzeitige Siliconcompiler erfüllen, sind sehr unterschiedlich. Sie reichen von Speziallösungen für ein eingeschränktes Schaltkreisspektrum bis zu Universallösungen, die jedoch oft nur eingeschränkte Entwurfsqualität liefern. Auch der Automatisierungsgrad solcher Systeme ist recht unterschiedlich. Gegenüber dieser Vielfalt innerhalb des gesamten Problemkreises kristallisieren sich auf unterem Niveau (eigentliche Umsetzung ins Silizium, d. h. Maskenlayout) zwei wesentliche Strategien heraus:

- Elektrikplan auf Transistorniveau
- System von Baublöcken (Moduln), die allesamt parametrisierbar sind und deren aktuelle Parameterbelegung automatisch abgeleitet wird.

Auf die Umsetzung eines detaillierten Transistorplanes ins Feinlayout soll hier nicht näher eingegangen werden. Den einzelnen Moduln hinterliegt im konkreten Fall ein parametrisierbares Modell, das funktionelle und strukturelle Aspekte gleichermaßen enthält. Ausgehend vom strukturellen Anteil des Modells kann die Umsetzung ins Feinlayout neben symbolischen Möglichkeiten (z. B. Sticks) oder Layoutsprachen auch über Leafcell-Systeme erfolgen. Letzteres bietet sich vor allem bei Moduln mit sehr hohem Wiederholgrad (wie ROM, RAM oder PLA) an.

Die Vorgehensweise soll anhand einer einfachen Grundstruktur der PLA erläutert werden (Bild 1).

2. Problemstellung und grundlegende Begriffe

Unter *Leafcell* soll hier die nicht weiter unterteilte Layoutbeschreibung (polygonales Layout in üblichen Formaten) eines Funktionselementes verstanden werden (Layoutfragment), das zusätzlich noch mit weiteren Informationen für die Symbolgenerierung, die mögliche Potentialbelegung und die Reihbarkeit versehen ist. Eine solche Leafcell ist im allgemeinen allein nicht funktionsfähig, da die Leafcells in der Regel elektrisch unvollständig (z. T. nur ein Transistor oder nur Teile davon) und auch geometrisch unvollständig sind; das heißt, daß bestimmte Polygone am Rand der Zelle enden und erst durch Anfügen der nächsten Leafcell zu einer entwurfsregel- und funktionsgerechten Gesamtfigur ergänzt werden. Die wichtigste Grundforderung an eine Leafcell ist die lückenlose Reihbarkeit der Zelle, die überhaupt erst die Blockgenerierung auf diesem Wege gestattet. Diese Reihbarkeit ist im vorliegenden Fall an eine rechteckige Grundstruktur der Zellen gebunden. In komfortableren Systemen könnte davon jedoch mit entsprechend hohem Aufwand auch abgewichen werden. Die Problemstellung für den Leafcellgenerator besteht demnach im lückenlosen Aneinanderreihen der Leafcells, so daß ein entwurfsregel- und funktionsgerechter Layoutblock entsteht, in dem die Elementarfunktion der Zelle mehrfach realisiert ist.

3. Der Leafcellgenerator

Der Leafcellgenerator erzeugt Layouts, Blocksymbole und Verdrahtunginformatio-

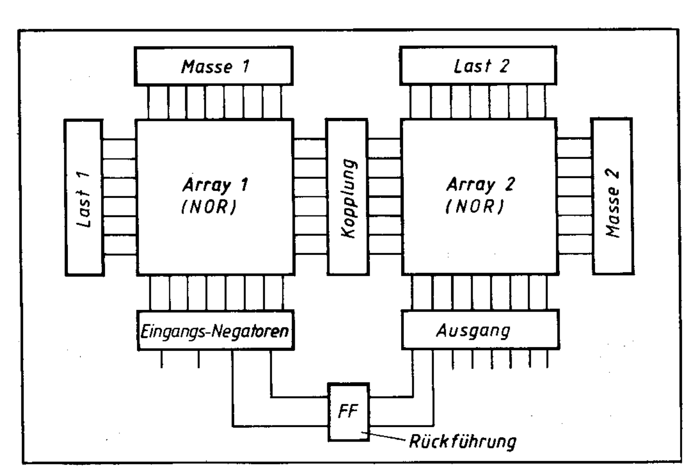


Bild 1 Struktur einer einfachen PLA

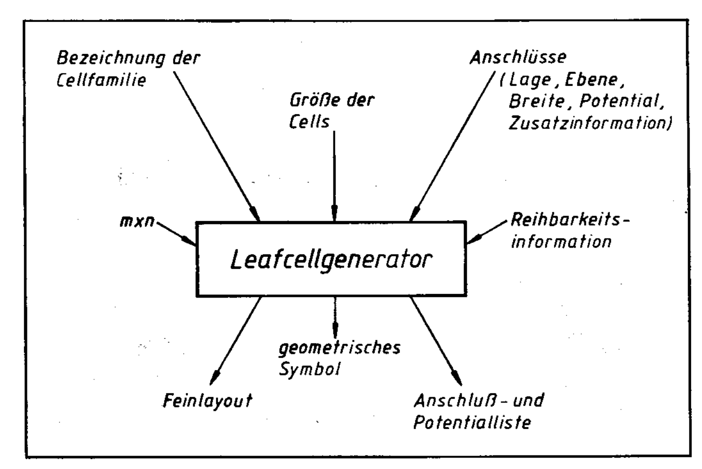


Bild 2 Ein- und Ausgänge des Leafcellgenerators

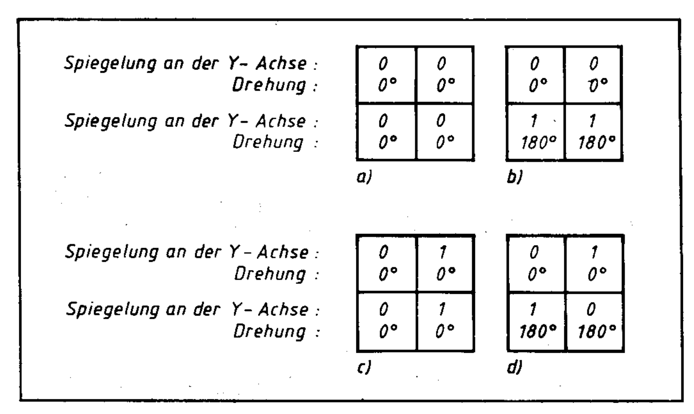


Bild 3 Vierergruppen von Zellen für unterschiedliche Lage der geteilten Strukturen
a) keine geteilten Strukturen
b) waagerechte geteilte Strukturen
c) senkrechte geteilte Strukturen
d) waagerechte und senkrechte geteilte Strukturen

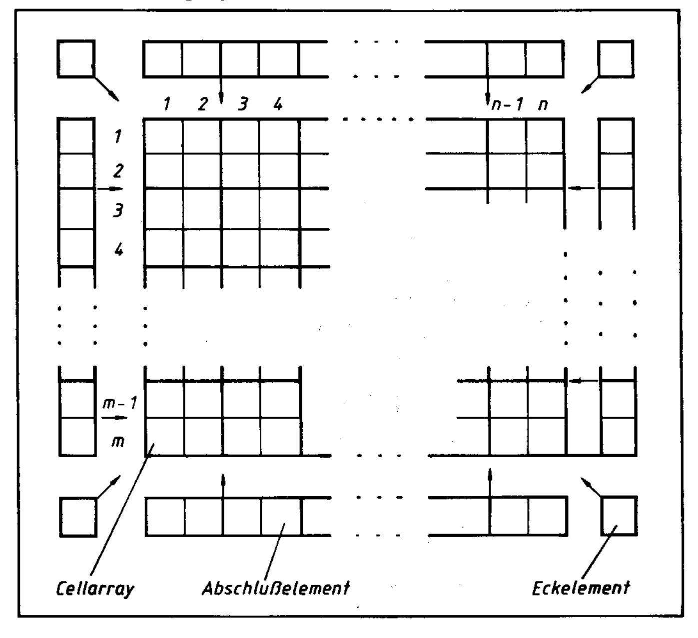


Bild 4 Leafcellarray, bestehend aus mehreren Vierergruppen und Anschlußelementen

nen für reguläre Funktionsblöcke, wobei im einzelnen folgende Aufgaben zu lösen sind (vergleiche auch Bild 2):

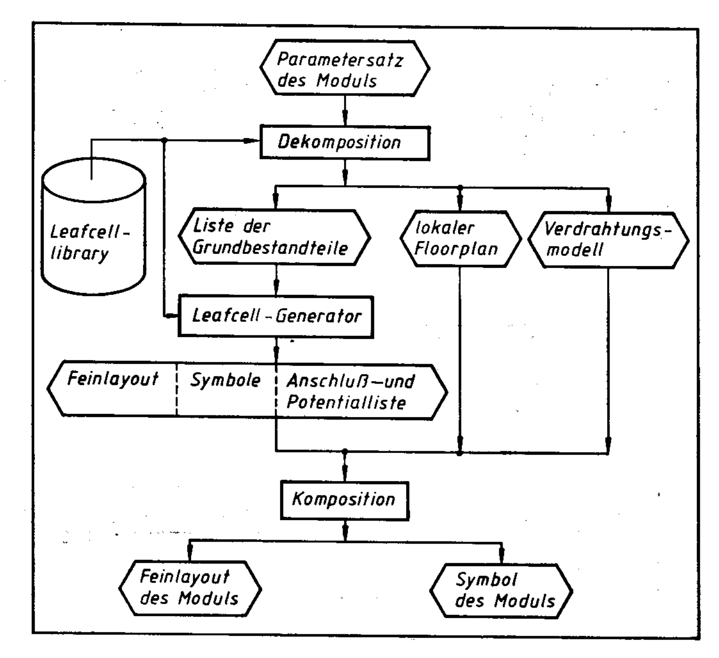
- Generierung eines Cellarrays mxn (m, n > 0; falls m, n > 1, dann m, n gerade)
- Zusammensetzung der Zellen so, daß geteilte Strukturen ergänzt werden (Drehung und Spiegelung einzelner Zellen im Array entsprechend besonderer Eingangsinformationen)
- Abschluß des Arrays an den Rändern mit sogenannten Abschlußelementen, so daß ein entwurfsregelgerechtes Gesamtlayout entsteht
- Da der Generator insbesondere für reguläre Strukturen eingesetzt wird, also auch für PLA-, ROM- und Decoderfelder, ist es sinnvoll, die Boolesche Belegung der Felder mit zu verarbeiten und demzufolge zwei zwar von der Randgestaltung gleiche, im Inneren aber verschiedene Leafcells (entsprechend „0" und „1") zu verwenden.

Für Spezialanwendungen kann es außerdem notwendig sein, neben dem Setzen von Aufrufpunkten für die Zellen auch bestimmte technologische Ebenen im Layout mit Figuren zu belegen. Diese Aufgabe ist jedoch sehr problemspezifisch und soll deshalb hier nur kurz erwähnt sein.

- Erzeugung eines maßstäblichen, rechteckigen Blocksymbols mit allen Außenanschlüssen
- Aufbau der blockspezifischen Potentialliste (Zuordnung der Potentiale zu den Anschlüssen).

Wie bereits erwähnt, sind Leafcells in der Regel geometrisch unvollständig. Das kommt daher, daß auf Grund der Regularität oft eine gemeinsame Nutzung bestimmter Layoutstrukturen (Masse- und Versorgungsleitungen, Kontaktfenster, aktive Gebiete usw.) durch benachbarte Zellen möglich wird. Das führt in den meisten Fällen dazu, daß die benachbarten Zellen gedreht bzw. gespiegelt aneinandergereiht werden. Der Leafcellgenerator arbeitet deshalb prinzipiell mit einer Vierergruppe von Zellen, die alle möglichen Varianten abdeckt (vgl. Bilder 3a bis 3d). Die Ergänzung der geteilten Strukturen durch Drehung und Spiegelung ist natürlich nur im Inneren des Leafcellblockes möglich, wogegen an den Außenkanten nach wie vor geteilte Strukturen auftreten können. Diese müssen durch spezielle Abschlußelemente, die ebenfalls zum Leafcellsystem gehören, ergänzt werden. Der Leafcellgenerator liefert somit einen Layoutblock, der geometrisch und elektrisch vollständig ist und den Entwurfsregeln genügt (Bild 4).

Dr.-Ing. Frank Lenke (29) studierte von 1977 bis 1982 an der Technischen Universität Dresden und diplomierte mit einer Arbeit über einen Entwurfsplatz für Multichip-Hybridschaltkreise. Anschließend Forschungsstudium an der TU Dresden und 1985 Promotion mit einer Arbeit über Redundanznutzung in hochintegrierten DRAMs.
Seit 1979 ist Dr. Lenke wissenschaftlicher Mitarbeiter im Zentralinstitut für Kybernetik und Informationsprozesse der AdW der DDR und befaßt sich derzeit mit Generatoren für reguläre Strukturen sowie dem Layoutentwurf und der Layoutoptimierung von Steuerwerken.



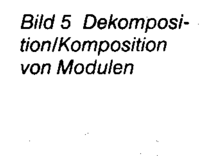
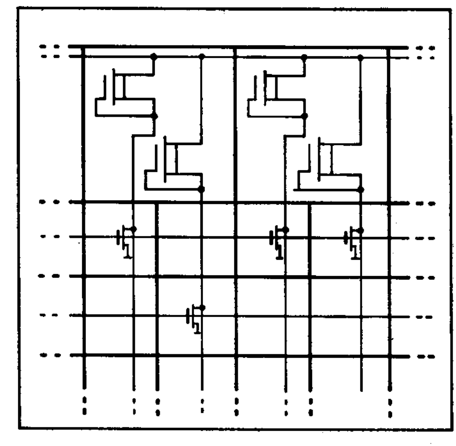

Bild 5 Dekomposition/Komposition von Modulen

Bild 6 Kombination von Eingangsnegatoren mit PLA-Zellen


Bild 7 Kombination von Lastelementen mit PLA-Zellen

4. Dekomposition/Komposition von Moduln

Ziel der strukturellen Dekomposition eines Moduls ist dessen Zerlegung in Grundbestandteile. Unter Grundbestandteilen sollen Funktionselemente verstanden werden, die – eine Elementarfunktion (ggf. mehrfach) realisieren (z. B. Speicherarrays, Treiberketten, Taktgeneratoren, Registerblöcke) – als fertiger Layoutblock einschließlich Symbol vorliegen oder aus gleichartigen Zellen (Leafcells) aufgebaut werden oder als Quelltext in einer Layoutsprache vorliegen.
Für sehr komplizierte Module ist auch eine Dekomposition über mehrere Hierarchiestufen denkbar, das heißt, der Modul wird nicht in Grundbestandteile, sondern in Submodule zerlegt, die wiederum den Dekompositionsalgorithmen unterworfen werden. Im vorliegenden Fall sollen die Grundbestandteile prinzipiell als Leafcellarrays (entsprechend dem eben beschriebenen Vorgehen) betrachtet werden. Demzufolge nimmt der Leafcellgenerator hier auch einen festen Platz bei der Umsetzung ins Feinlayout ein (Bild 5).

4.1. Liste der Grundbestandteile

Jeder Modulgenerator verwaltet eine Menge von Grundbestandteilen GB:
Nicht jede Realisierung dieses hier abstrakt beschriebenen Moduls benötigt alle Grundbestandteile. Anhand eines aktuellen Parametersatzes werden die benötigten Elemente aus GB ausgewählt, das heißt, der Modul wird konfiguriert. Im hier dargelegten Fall der Leafcellgenerierung muß jedem Element aus GB mindestens eine Leafcell zugeordnet werden können und es muß aus der Cell-Library abrufbar sein. Für das im Bild 1 dargestellte PLA-Modell würde sich folgende Menge GB ergeben:

GB_{PLA} = { Eingangsnegatoren,
Array – 1,
Last – 1,
Masse – 1,
Kopplung,
Array – 2,
Last – 2,
Masse – 2,
Ausgang,
Rückführung }

In diesem einfachen Fall wäre die Modulgenerierung relativ einfach, da fast alle Elemente von GB für den Modul PLA benötigt werden. Lediglich die Rückführung ist in diesem Fall variabel und muß über Parameter gesteuert werden. In komplizierten Modulen ist jedoch die Mächtigkeit von GB wesentlich

größer als die Anzahl der durchschnittlich benötigten Grundbestandteile.
Ein weiteres Problem der Modulkonfigurierung besteht darin, daß die Elemente von GB ebenfalls wieder als Mengen betrachtet werden können, deren Mächtigkeit größer als 1 sein kann. Im obigen Beispiel PLA könnte z. B. die Menge AUSGANG folgendermaßen aufgebaut sein:

AUSGANG = { nichtnegierende Treiber,
negierende Treiber,
setzbare nichtnegierende Treiber,
setzbare negierende Treiber,
... }

Im Anschluß an die Modulkonfigurierung stehen Leafcells fest, aus denen die Grundbestandteile aufgebaut werden. Im nächsten Schritt werden die einzelnen Grundbestandteile konfiguriert, das heißt, die Größe der Zellarrays mxn (entsprechend Bild 4) wird festgelegt. Dabei haben die inneren Verarbeitungsbreiten, die I/O-Gestaltung und ggf. weitere Parameter Einfluß auf die Konfigurierung. Andererseits nimmt auch die Zellgröße selbst Einfluß auf die Arraydimension mxn, da sowohl aus geometrischen wie auch aus funktionellen Gründen der Fall eintreten kann, daß eine Leafcell des einen Grundbestandteils jeweils mit mehreren Leafcells eines anderen Grundbestandteils verbunden ist. Am Beispiel PLA trifft das einerseits auf Eingangsnegatoren zu, die (funktionell bedingt) jeweils zwei Zellen des Eingangsfeldes ansteuern (Bild 6). Andererseits ist ein ähnliches Vorgehen z. B. bei statischen Lastelementen denkbar. Da die Depletiontransistoren oft breiter sind als die Zellen der PLA-Felder, ist hier eine versetzte Anordnung notwendig, so daß in diesem Fall aus strukturellen Gründen ein Lastelement auf beispielsweise zwei Reihen des PLA-Feldes wirkt (Bild 7).
Die zu den – in der Menge GB enthaltenen – Grundbestandteilen gehörenden Leafcells müssen in vielen Parametern, wie Größe, Anschlußbelegung, Technologie, Schaltungstechnik usw., aufeinander abgestimmt werden. Soll der Modulgenerator eine hohe Flexibilität erreichen, so muß er mehrere Mengen von Grundbestandteilen (GB_1, GB_2, ...) verwalten. Im Beispiel PLA könnten das sein:
- schnelle oder langsame Varianten
- dynamische oder statische Technik
- kleine, mittlere oder große Felder
- Variation innerhalb einer Technologieklasse (z. B. NMOS-Technologien mit unterschiedlicher Stegbreite).

Der Wechsel zwischen Technologieklassen (z. B. zwischen NMOS, CMOS oder ECL) ist meist nicht durch den einfachen Austausch der Mengen GB_i zu erreichen, sondern erfordert Eingriffe in den Generator bzw. in das Strukturmodell, die hier nicht näher diskutiert werden sollen.

4.2. Lokaler Floorplan

Der lokale Floorplan gibt auf der Stufe der Dekomposition die relative Lage der (im vorliegenden Fall rechteckigen) Grundbestandteile zueinander an. Relativ deshalb, weil die Größen der Grundbestandteile und die Anschlußbelegungen hier noch nicht bekannt sind. Die konkreten Lageinformationen liegen dann erst auf der Stufe der Komposition vor. Den lokalen Floorplan kann man auch als allgemeines Strukturmodell des Moduls interpretieren. Die prinzipielle geometrische Anordnung der Grundbestandteile ergibt sich aus deren Funktion. Für übersichtliche, reguläre Anordnungen, wie etwa die PLA nach Bild 1, ergeben sich auch entsprechend einfache Regeln für den Floorplan, zum Beispiel:
ordne die Eingangsnegatoren unmittelbar an der Eingangsseite der Matrix 1 an, wogegen an der gegenüberliegenden Seite dieser Matrix die Masseelemente (Masseleitungen) plaziert werden; usw.
Diese hier sehr einfachen Regeln werden bei kompliziert strukturierten Modulen recht aufwendig, da viele Nebenbedingungen beachtet werden müssen. Dennoch ergeben sich drei wesentliche *Vorteile:*

- Innerhalb des Moduls kommt kein Placementalgorithmus zur Anwendung, da die Anordnung der Grundbestandteile aus funktionellen Gesichtspunkten vorgenommen wird und unter Ausnutzung vorhandener Freiheitsgrade im Generator implementiert ist.

- Die fest vorgegebene (in der Regel lückenlose) Aneinanderreihung der Grundbestandteile ist eine wesentliche Voraussetzung für die implizite Verdrahtung der Grundbestandteile untereinander (wireing by abutment), das heißt, zu verbindende Außenanschlüsse der Grundbestandteile stehen sich im Feinlayout direkt gegenüber und werden durch Aneinanderschieben der Grundbestandteile implizit verbunden. (Dieses Verfahren setzt natürlich neben dem geeigneten Strukturmodell auch die Paßfähigkeit der Layouts voraus.)

- Da nur die relative Lage der Grundbestandteile in Form parametrisierter Gleichungen beschrieben wird, können Freiheitsgrade in der Anordnung durch Drehen, Spiegeln, Vertauschen oder Verschieben von Grundbestandteilen ausgenutzt werden. Allein für das einfache Strukturmodell der PLA nach Bild 1 ergeben sich 16 Möglichkeiten für die Anordnung der Grundbestandteile.

Bild 8 zeigt einen Ausschnitt des parameterbehafteten Floorplans für die PLA, woraus sich z. B. folgende Plazierungsgleichung für die Eingangsmatrix der PLA ergibt (Punkt P1):

$$P1X = RL + X_I + AX_1 + 1/2X_{III}$$
$$P1Y = RU + Y_{IV} + AY_1 + 1/2Y_{III}$$

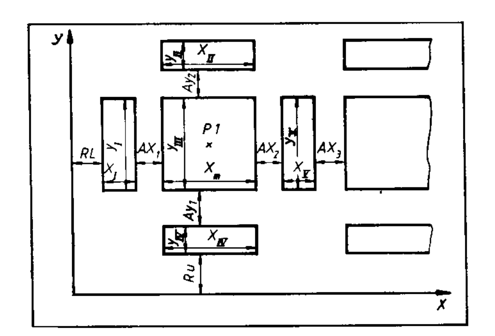


Bild 8 Ausschnitt aus dem lokalen Floorplan einer PLA

4.3. Das Verdrahtungsmodell

Die Verdrahtung im Inneren des Moduls teilt sich in zwei Problemkreise auf:
– Verdrahtung durch Aneinanderreihung der Grundbestandteile (wireing by abutment)
– freie Verdrahtung (interaktiv oder mittels Router) zwischen den Grundbestandteilen und zu den Außenanschlüssen des Moduls.

Die Verdrahtung durch Aneinanderreihung stellt hohe Ansprüche an die Feinlayouts der Grundbestandteile. Die zu verbindenden Anschlüsse müssen sich paarweise gegenüberstehen, in ihrer Breite und Layoutebene übereinstimmen. Außerdem dürfen an den zu reihenden Seiten der Grundbestandteile keine Anschlüsse für die freie Verdrahtung sein, da zwischen den gereihten Grundbestandteilen kein Verdrahtungskanal im üblichen Sinne entsteht. Diese Art der Verquickung von Plazierung und Trassierung unter Ausnutzung spezieller Layoutgestaltung wird im konventionellen Schaltkreisentwurf schon seit Jahren angewendet, insbesondere bei Array- und Bitslicestrukturen. Für die Generierung stellt sie auf Grund wachsender Regularität im VLSI-Bereich eine wichtige Grundlage für schnelle automatische Entwurfssysteme dar.
Die freie Verdrahtung wird interaktiv oder mit einem Router durchgeführt. In beiden Fällen sind die zur Verfügung zu stellenden Ausgangsdaten:
– Verdrahtungsnetz
– Anschlußpositionen und -ebenen
– Verdrahtungsräume (-kanäle)
– Verdrahtungsebenen (ggf. mit Vorzugsrichtungen).

Diese Informationen sind in der Regel wesentlich umfangreicher als die Informationen für die Verdrahtung durch Aneinanderreihung. Demgegenüber sind aber auch die Freiheitsgrade für die Plazierung der Grundbestandteile und die Anordnung der Außenanschlüsse wesentlich größer.

5. Schlußbemerkung

Die vorgestellte Methodik zur Layoutgenerierung komplexer Strukturen wurde an mehreren Beispielen übersichtlicher ROM- und PLA-Strukturen erfolgreich getestet. Durch geeignete Schnittstellen im Feinlayout- und Symbolbereich ist ein Zusammenfügen mit herkömmlich entworfenen Modulen problemlos möglich.

Literatur
/1/ Eckhardt, D., u. a.: Studie Siliconcompiler. ZKI der ADW, Dresden, 1986; internes Material
/2/ Lenke, F.: Konzeption eines Leafcellsystems. Fachtagung „Computer- und Mikroprozessortechnik '86" Berlin, 1986

⊠ KONTAKT ☏

Akademie der Wissenschaften der DDR, Zentralinstitut für Kybernetik und Informationsprozesse, Institutsteil Dresden, Haeckelstr. 20, Dresden, 8027; Tel. 463 31 64.

Stand der Entwicklung von Festwertspeichern

Heiko Bialozyt
VEB Mikroelektronik
„Karl Marx" Erfurt

Vorbemerkungen

Mit dem bisher im Kombinat Mikroelektronik entwickelten Sortiment an Prozessoren – zum Teil mit Festwertspeichern – stehen dem Anwender zur Lösung seiner Aufgaben zahlreiche Schaltkreistypen zur Verfügung. Tafel 1 gibt eine Übersicht über die bereitstehenden Prozessorschaltkreise.
Außer der Notwendigkeit der Verbesserung der Prozessoren ist es unumgänglich, die Speicher weiterzuentwickeln. Einerseits sind stetig größer werdende Datenmengen zu verarbeiten, wofür immer größere RAM-Bereiche notwendig werden, andererseits müssen dazu immer mehr und größere Programme und Programmsysteme fest im Rechner installiert werden. Dies läßt sich auf zwei verschiedenen Wegen realisieren.
Bis zu einer bestimmten Grenze kann die Aufstockung des Speicherbereiches durch den Einsatz von zusätzlichen Schaltkreisen erzielt werden.
Stehen jedoch nur begrenzt Raum, Energie bzw. ökonomische Mittel zur Verfügung, muß zu moderneren, höherintegrierten Speichern übergegangen werden.

Festwertspeicher aus dem VEB MME

Bild 1 gibt eine Übersicht über die Typen und Pinbelegungen der Speicherschaltkreise, die derzeit im VEB Mikroelektronik „Karl Marx" Erfurt (MME) hergestellt werden.
Zwei Typen dieser Reihe stellen EPROM-Speicher dar (U2716C und U2732C). Diese Speicher besitzen ein Fenster in ihrem Gehäusedeckel, wodurch die in Floating-Gates gespeicherte Information wieder gelöscht werden kann. Hierfür sind die Schaltkreise mit ultraviolettem Licht zu bestrahlen. Auf diese Weise können die Speicher mehrfach und für verschiedene Daten verwendet werden. Haupteinsatzgebiet dieser Schaltkreise sollte wegen der hohen Kosten die Software- bzw. Geräteentwicklung sein. Für den Einsatz in Geräten, die nur in sehr geringer Stückzahl produziert werden, ist der Einsatz von EPROM ebenfalls vorzusehen, da die Herstellung eines speziellen Bitmusters nicht vertretbar wäre.
Werden mehr als 500 Stück eines Gerätes im Jahr hergestellt, ist die Verwendung von PROM-Schaltkreisen bereits günstig möglich.
Zu dem EPROM U2716C bietet der VEB MME einen pinkompatiblen PROM mit der Bezeichnung U2616D. Dieser Typ kann aber aufgrund des Plastgehäuses billiger produziert werden.
Nach Auslösen einer Bitmusterbestellung, der die Masterbauelemente beizufügen sind, wird der Schaltkreis vom Hersteller programmiert. Wird der PROM vom Anwender weiter- oder überprogrammiert, erlischt gleichzeitig jeder Garantieanspruch.
Da dieser Schaltkreis das gleiche Chip wie der EPROM enthält, werden hinsichtlich Lebensdauer und Datensicherheit nur die Werte des EPROM erreicht. Sollen höhere Werte garantiert werden, sind maskenprogrammierte Speicher einzusetzen. Bei diesen ROM-Speichern wird die zu speichernde Information direkt während der Herstellung des Chips in Form entsprechender Verbindungen in die Struktur der Speichermatrix eingebaut. Ein Ändern der einmal gespeicherten Information ist damit nicht mehr möglich. Gleichzeitig bedeutet die Herstellung einer Maske für die Produktion des gewünschten ROM-Bitmusters den größten Aufwand zur Programmierung eines Schaltkreises sowohl in Hinsicht auf die benötigte Zeit als auch aus ökonomischer Sicht. Zur Bestellung und Produktion eines neuen Bitmusters ist dabei der in Tafel 2 dargestellte Ablauf einzuhalten.
Im Gegensatz zu den EPROM- und PROM-Speichern kann also hier die Produktion erst dann beginnen, wenn Änderungen im Bitmuster nicht mehr möglich sind. Außerdem vergehen von der Bestellung bis zur Produktion der ersten Schaltkreise eines neuen Bitmusters rund sechs Monate. Die Maskenkosten sind auf den Grundpreis des Schaltkreistyps noch aufzuschlagen. Damit wird der Einsatz von maskenprogrammierten ROM-Speichern erst ab Stückzahlen größer 5000 rentabel. Ebenfalls ist eine ökonomisch günstige Produktion erst oberhalb dieser Grenze realisierbar, weshalb 5000 Stück pro Jahr als Mindeststückzahl für die Bestellung von Bitmustermasken gefordert wird.
Aus diesen Gründen können drei Einsatzkategorien für Festwertspeicher unterschieden werden, denen jeweils eine Gruppe von Speichern zugeordnet ist (siehe dazu Tafel 3).

Tafel 2 Ablauf für die Herstellung von ROM-Bitmustern

Nr.	Schritt	Bemerkungen
1.	Bitmusterbestellung	– Der gewünschte Datensatz ist auf einem Datenträger entsprechend dem gültigen Werkstandard zu übergeben. – Gleichzeitig ist die Festlegung der Wirksamkeit der programmierbaren CS-Eingänge vorzunehmen.
2.	Übernahme	– Das Bitmuster wird auf einen Datenträger des Herstellers übernommen. – Ein Kontrollausdruck wird dem Besteller zugesandt.
3.	1. Kontrolle	– Der Kontrollausdruck ist auf Fehler zu untersuchen. – Bei Auftreten von Fehlern ist ein neuer Datensatz zu übergeben. – Die Richtigkeit ist zu bestätigen.
4.	Bearbeitung der Daten	– Bei Richtigkeit des übernommenen Datensatzes wird mit der Bearbeitung begonnen. – Solange Fehler auftreten, wird 2 und 3 wiederholt.
5.	Herstellung der Maske	– Die Informationen des Datensatzes werden in die entsprechenden Verbindungen auf dem Chip übersetzt. Es entsteht eine Maske.
6.	Produktion	– Auf Grundlage der neuen Maske wird der Schaltkreis produziert. – Nach Kontrolle kann dieser an den Anwender ausgeliefert werden.

Tafel 1 Mikroprozessorschaltkreise des MME

Typ	Breite	cp_{MAX} (MHz)	Speicherbereich (KByte)	Bemerkungen
UB880	8Bit	2,5	64	durch I/O-IS unterstützt
UA880	8Bit	4,0	64	(PIO, SIO, CTC, DMA)
U881	8Bit	4,0	2 (ROM intern)	EMR
U882			62 (ROM/RAM extern für Programm)	mit PIO, TIMER, SIO – zwei Varianten:
U883			62 (RAM extern für Daten)	* POWER-DOWN * interner Taktgenerator
U884	8Bit	4,0	4 (ROM intern)	– Entwicklungsversion
U885			60 (ROM/RAM extern für Programm) 60 (RAM extern für Daten)	– 128-Byte-RAM für Register
U8001	16Bit	4,0	16 MByte (segmentiert)	Unterstützung der Speicherverwaltung U8010
U8002	16Bit	4,0	64 (1 Segment)	nichtsegmentiert

Tafel 3 Einsatzkategorien und Speichergruppen

Speichergruppe	Einsatzkriterien
EPROM	– häufige Änderungen in der Software – sehr geringe Mengen bei der Produktion (500) • *Haupteinsatz in Forschung und Entwicklung*
PROM	– Produktionsmengen von 500 bis 5000 Stück – kurzfristig (2 Monate) Änderungen bei laufender Produktion des Schaltkreises möglich – geringere Kosten als gleichwertiger EPROM – voll pinkompatibel zum EPROM • *Haupteinsatz in der Kleinserienfertigung*
ROM	– hohe Produktionsstückzahlen (5000) – gesicherte Software, d. h., keine bzw. kaum Änderungen notwendig – geringste Kosten bei entsprechend hoher Stückzahl – größte Datensicherheit und Lebensdauer • *Haupteinsatz in der Großserienproduktion*

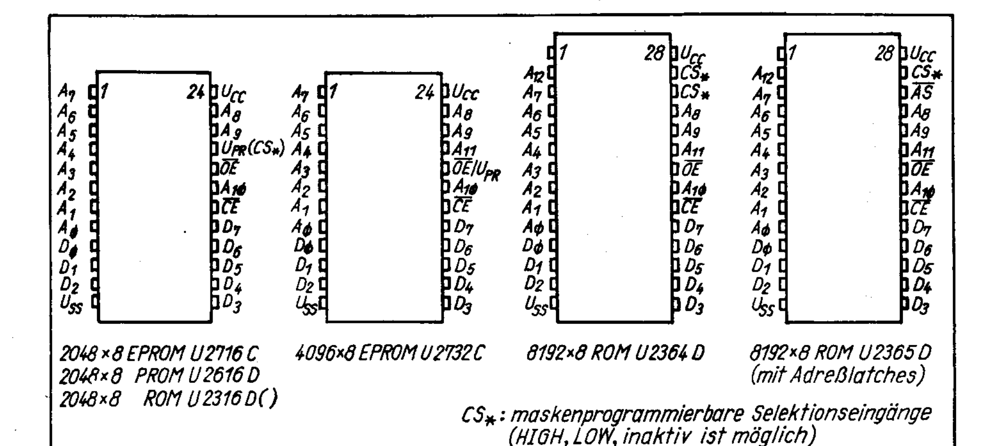
Tafel 4 *Byte-Wide-Standard bis 4K × 8*

alle Typen	24polig Pin	Pin	RAM	ROM 2316	2K×8 PROM 2616	EPROM 2716	E^2PROM 2816	4K×8 EPROM 2732
A_7	1	24	U_{CC}	U_{CC}	U_{CC}	U_{CC}	U_{CC}	U_{CC}
A_6	2	23	A_8	A_8	A_8	A_8	A_8	A_8
A_5	3	22	A_9	A_9	A_9	A_9	A_9	A_9
A_4	4	21	/WE	CS*	U_{PR}	U_{PR}	U_{PR}	A_{11}
A_3	5	20	/OE	/OE	/OE	/OE	/OE	/OE
A_2	6	19	A_{10}	A_{10}	A_{10}	A_{10}	A_{10}	A_{10}
A_1	7	18	/CE	/CE	/CE	/CE	/CE	/CE
A_0	8	17	D_7	D_7	D_7	D_7	D_7	D_7
D_0	9	16	D_6	D_6	D_6	D_6	D_6	D_6
D_1	10	15	D_5	D_5	D_5	D_5	D_5	D_5
D_2	11	14	D_4	D_4	D_4	D_4	D_4	D_4
U_{SS}	12	13	D_3	D_3	D_3	D_3	D_3	D_3

Tafel 5 *Byte-Wide-Standard 8K × 8 bis 32K × 8*

alle Typen	28polig Pin	Pin	RAM	ROM 2364	8K 2365	EPROM 2764	16K 27128	32K 27256
U_{PR}	1	28	U_{CC}	U_{CC}	U_{CC}	U_{CC}	U_{CC}	U_{CC}
A_{12}	2	27	/WE	CS*	CS*	/PGM	/PGM	A_{14}
A_7	3	26	N.C.	CS*	/AS	N.C.	A_{13}	A_{13}
A_6	4	25	A_8	A_8	A_8	A_8	A_8	A_8
A_5	5	24	A_9	A_9	A_9	A_9	A_9	A_9
A_4	6	23	/WE	CS*	U_{PR}	U_{PR}	U_{PR}	A_{11}
A_3	7	22	/OE	/OE	/OE	/OE	/OE	/OE
A_2	8	21	A_{10}	A_{10}	A_{10}	A_{10}	A_{10}	A_{10}
A_1	9	20	/CE	/CE	/CE	/CE	/CE	/CE
A_0	10	19	D_7	D_7	D_7	D_7	D_7	D_7
D_0	11	18	D_6	D_6	D_6	D_6	D_6	D_6
D_1	12	17	D_5	D_5	D_5	D_5	D_5	D_5
D_2	13	16	D_4	D_4	D_4	D_4	D_4	D_4
U_{SS}	14	15	D_3	D_3	D_3	D_3	D_3	D_3

Bild 1 ▼ Verfügbare Festwertspeicher aus dem VEB MME



Byte-Wide-Standard

Um ein möglichst unkompliziertes Übergehen von einem Speichertyp auf einen anderen sowie die leichte Austauschbarkeit der einzelnen Typen einer Größe zu garantieren, ist es notwendig, alle Schaltkreise mit einer weitestgehend identischen Pinbelegung auszustatten.

Hierfür ist international der von JEDEC herausgegebene Industriestandardvorschlag üblich. Um auch auf internationalen Märkten unsere Schaltkreise anbieten zu können, wird dieser Standard auch für die Schaltkreise des VEB MME angewendet. Damit können byteorganisierte Speicher (RAM, ROM, EPROM, E^2PROM) mit Speicherkapazitäten von 2K × 8 bis 64K × 8 in 24- bzw. 28poligen Gehäusen mit geringsten Hardwareänderungen verwendet werden. Tafel 4 zeigt die Belegung für Speicher bis 32 KBit Kapazität.

Speicher größerer Kapazität (2K × 8)

Wie in Bild 1 zu sehen ist, stehen nur für 2K × 8-organisierte Speicher alle drei Gruppen (EPROM, PROM, ROM) bereit.

Da jedoch auch größere Speicher mit der momentan beherrschbaren Technologie herstellbar sind, werden diese Möglichkeiten voll ausgeschöpft. Dadurch können dem Anwender auch 4K × 8-EPROM- und 8K × 8-ROM-Speicher angeboten werden. Tafel 5 zeigt die Belegung für Speicher bis 512 KBit Kapazität.

Dipl.-Ing. Heiko Bialozyt (25) studierte von 1980 bis 1983 an der Offiziershochschule „Franz Mehring" in Kamenz. Er ist seit 1985 im VEB Mikroelektronik „Karl Marx" Erfurt in der Mikroprozessorapplikation tätig und diplomierte 1986 an der Technischen Hochschule Ilmenau.
Sein Aufgabengebiet ist das Speichersortiment und die 16-Bit-Mikroprozessortechnik.

Die Speicherkapazität von 4K × 8 (U2732C) stellt hierbei nur eine zeitweilige Größe bis zur Produktion von 8K × 8-EPROM und PROM-Speichern dar. Aus diesem Grunde sollte bei der Konzipierung von ROM-Bereichen ab 8 KByte bereits jetzt auf diese Typen orientiert werden. Der vorläufige Einsatz von U2732C ist durch die Anwendung des JEDEC-Standards unkompliziert möglich.

Bei 8K × 8-RAM-Speichern ist U_{PR} (Pin 1) nicht angeschlossen.

Mit einer Kapazität von 8K × 8 werden vom VEB MME zwei verschiedene Typen angeboten. Der eine Typ (U2364D) besitzt zwei maskenprogrammierbare Chip-Select-Eingänge. Diese können auf eine der drei Möglichkeiten HIGH-, LOW-aktiv oder inaktiv eingestellt werden. Damit ist die Zusammenschaltung von bis zu vier Schaltkreisen ohne Dekoder möglich. Die andere Variante (U2365D) besitzt an Stelle des zweiten CS*-Einganges nun die Möglichkeit, die internen Adreßlatches zu steuern. Auf diese Art und Weise kann der Speicher direkt an einem gemultiplexten Adreß-/Datenbus arbeiten.

Ein EPROM/PROM der Kapazität 8K × 8 wird entwickelt. Damit stehen auch für diese Speichergröße alle drei Gruppen bereit.

Intelligente Programmierung von EPROM-Speichern

Mit zunehmender Integration sind auch immer größere Datenmengen zu entwickeln und zumindest zeitweilig auf EPROM zu programmieren. Bei Verwendung der bisher üblichen Algorithmen steigt also auch die Programmierzeit proportional mit der Kapazität der Speicher. Wie aus Tafel 6 zu ersehen ist, müssen damit für einen 8K × 8-EPROM bereits sieben Minuten veranschlagt werden, da die Zellen mit minimal 45-ms-Impulsen programmiert werden.

Betrachtet man die Aufladung der einzelnen Zellen einmal genauer (siehe dazu Bild 2), so wird deutlich, daß die meisten Zellen bereits nach rund 8 ms programmiert sind.

Es ist also möglich, die Zeit zur Programmierung erheblich zu verkürzen, wenn diese für jede Zelle individuell ermittelt wird. Es gibt nun einen Algorithmus, der garantiert, daß eine EPROM-Zelle sicher ein LOW am Ausgang erzeugt und im gesamten Bereich von U_{CC} nicht auf logisch HIGH wechselt. Bild 3 zeigt die Kurven einer nichtprogrammierten und einer vollständig programmierten EPROM-Zelle. Die Kurven von unvollständig programmierten Zellen liegen dazwischen. So kann es zum Beispiel passieren, daß eine EPROM-Zelle, die unvollständig programmiert ist, bei U_{CC} = 5 Volt zwar ein LOW liefert, aber bei Ansteigen von U_{CC} auf HIGH kippt.

Der Algorithmus beginnt also mit dem Einstellen von U_{CC} = 6 Volt. Das ist höher als der normale Arbeitsbereich, aber notwendig, um den Programmierbereich zu sichern. Anschließend wird die Programmierspannung von 25 (21) Volt eingestellt und iterativ mit 1-ms-LOW-Impulsen programmiert. Nach je-

Tafel 6 *Entwicklung der Programmierzeiten für EPROM*

Typ	Bytes	Zeit (min)
U552 (1702)	256	2,0
U555 (2708)	1024	1,5
U2716	2048	1,75
U2732	4096	3,5
(U2764)	8192	7,0

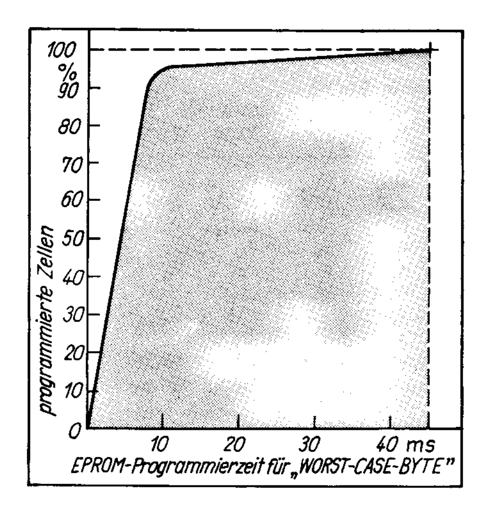


Bild 2 ◀ *Programmierverhalten von EPROM-Speicherzellen*

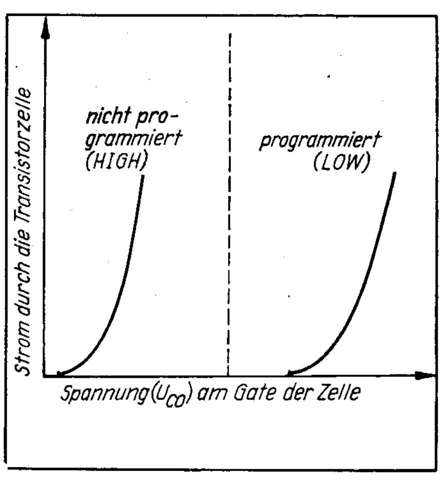


Bild 3 ◀ *Verhalten einer EPROM-Zelle im programmierten bzw. nichtprogrammierten Zustand*

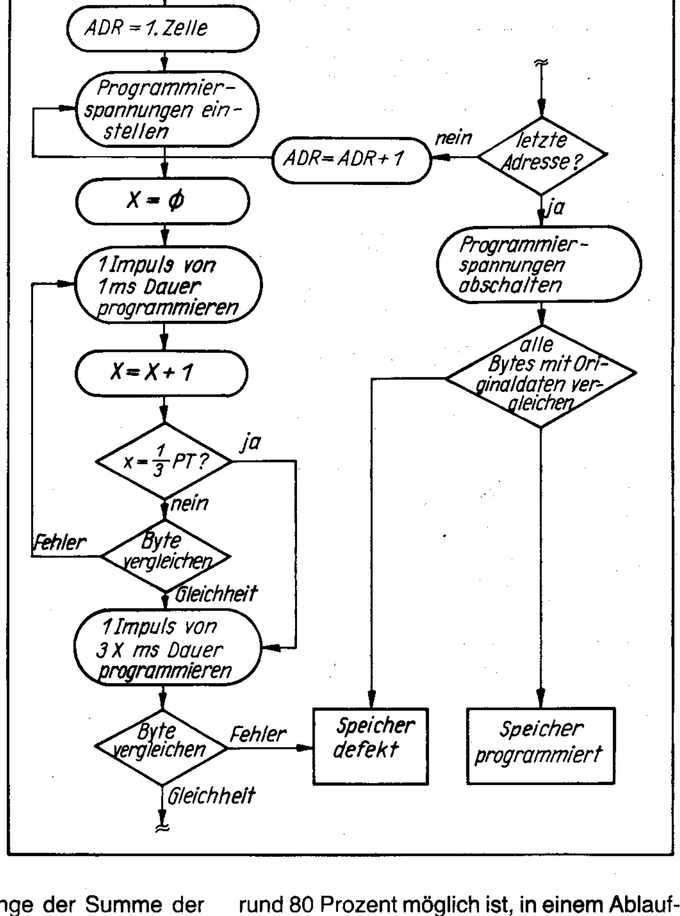


Bild 4 ▶ *Ablaufplan für „intelligente" Programmierung*

dem Impuls wird der Inhalt mit dem gewünschten Wert verglichen. Treten hierbei Fehler auf, wird der Programmier- und Testzyklus wiederholt. Nach 15 Impulsen wird die Zelle sofort mit 45 ms nachprogrammiert, da eine Verkürzung der Programmierzeit nicht mehr möglich ist. Sonst wird an diesem Punkt des Algorithmus noch ein Programmierimpuls der dreifachen Länge der Summe der bisher programmierten Impulse ausgeführt. Damit kann der längste Impuls 45 ms betragen. Für die meisten EPROM-Bytes genügen jedoch zwei bis drei 1-ms-Impulse, so daß diese Zellen mit 8 bis 12 ms programmiert sind. Bild 4 stellt diesen Algorithmus, mit dem eine Verkürzung der Programmierzeit um rund 80 Prozent möglich ist, in einem Ablaufplan dar.

☒ KONTAKT ☏

VEB Mikroelektronik „Karl Marx" Erfurt,
Abt. CEE 41, Erfurt, 5010, Tel. 5 10 76, App. 58
Rudolfstr. 47,

Neuer Superrechner

Suprenum 1 heißt der in Berlin (West) entwikkelte neue Superrechner mit einer Leistung von 5 Milliarden Operationen/Sekunde. Diese Geschwindigkeit erreicht der schnelle Computer des 1983 gegründeten Forschungszentrums für innovative Rechnersysteme und -technologie (First) der Technischen Universität und der Gesellschaft für Mathematik und Datenverarbeitung (GMD) durch die wirkungsvolle Zusammenschaltung zahlreicher Unterrechner und keineswegs nur durch schnellste Elektronik. Ausgenutzt wurde u. a. die hohe Packungsdichte der CMOS-Bauelemente.
Suprenum 1 verfügt über 256 parallel arbeitende Unterrechner, die jeweils auf einer Platine Platz finden. Der Computer ist in der Lage, große Datenmengen zu bewältigen, die zum Beispiel in der Strömungsphysik, für Entwicklungen im Flugzeug- und Automobilbau, in der Teilchenphysik und in der Meteorologie für die Wettervorhersage, aber auch bei Expertensystemen anfallen. Ein Prototyp soll im Frühjahr 1988 zum Einsatz kommen.

Computer versteht gesprochenen Text

Unter Beteiligung des Instituut foor Perceptie Onderzock, Eindhoven (Niederlande), gelang es den Firmen Siemens und Philips, das Spracherkennungssystem Spicos zu entwickeln. Der Benutzer muß bei diesem Dialogsystem Anfragen und Anweisungen nur noch ins Mikrofon sprechen. Das kürzlich fertiggestellte Funktionsmuster soll nicht nur einzelne gesprochene Wörter aus einem begrenzten Vokabular erkennen können, sondern den Bedeutungsgehalt vollständiger Sätze „verstehen".
Dadurch ist z. B. der verbale Zugriff in gewöhnlicher Umgangssprache auf Datenbanken möglich. Spicos ermöglicht das Verwalten von Aktennotizen, Briefen und ähnlichen Schriftsätzen. Das Vokabular umfaßt z. Z. ungefähr 1000 Wörter. Spicos soll auf Basis dieses Wortschatzes etwa 1,5 Billionen unterschiedliche Sätze erkennen können. Die gesprochenen Sätze müssen grammatikalisch korrekt sein.

Super-Chip

Der im Rahmen des staatlichen US-amerikanischen VHSIC-Projekts entwickelte neue Super-Chip mißt 35 × 35 mm und kann 18,8 Millionen aktive Elemente aufnehmen. Die Makrozellen des komplexen DSP (Digital Signal Processor) können unter anderem auf einen statischen RAM-Speicher mit einer Kapazität von 2 MBit zurückgreifen, ein doppelt so hoher Wert als bei den einzeln gefertigten 1-MBit-RAMs. Die Herstellung vollzieht sich auf CMOS-Basis. Mit „neuartiger automatischer Software-Konfiguration" soll auch dem Problem der hohen Ausschußquote von Superchips beigekommen worden sein. Mit Hilfe eingebauter Redundanz kann der Chip-Zustand effektiv in jeder Phase getestet werden; Defektstellen werden so umgangen. Mit der Entwicklung sind nunmehr vier der geplanten 29 Makrozellen der Chip-Gesamtschaltung des Projekts produktionsreif.

Anforderungsspezifikation und Modellbildung auf der Basis von Netzen

(Teil 1)

Dr. Stephan Fensch, VEB Kombinat Trikotagen, Karl-Marx-Stadt
Jürgen Lange, Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik

In vielen Publikationen zum Thema Anforderungsspezifikation wird darauf verwiesen, daß Softwarekosten und -qualität sowie die Einhaltung von Bereitstellungsterminen für Softwareprodukte auch beim Entwurf von Automatisierungssystemen wesentlich von der Güte der Anforderungsspezifikation bestimmt werden. Diese bildet die Ausgangsbasis für die gesamte weitere Entwicklung und muß deshalb durch Vollständigkeit, Widerspruchsfreiheit und Verträglichkeit gekennzeichnet sein. Solche Eigenschaften sind aber nur durch einen hohen Grad an Formalisierung der Spezifikation erreichbar. Im ersten Teil des Beitrags wird an Hand eines konkreten Beispiels gezeigt, wie mit Hilfe von systematisch entwickelbaren Netzmodellen dieser Grad erreicht werden kann, ohne an Verständlichkeit zu verlieren. Im zweiten Teil wird dann die rechnergestützte Simulation eines solchen Modells mit dem Programmpaket POSES (Pr/T-netzorientiertes Simulations- und Entwurfssystem) vorgestellt.

Einführung

Die zunehmende Komplexität moderner rechnerintegrierter Automatisierungssysteme (CIM) verlangt ein systematisches Entwurfsvorgehen mit hoher Effektivität. Durch immer bessere Entwicklungswerkzeuge, insbesondere in Form komfortabler Sprachkonzepte (PASCAL, MODULA2, ADA), verlagert sich die relative Fehlerhäufigkeit mehr und mehr in die Phase der Anforderungsspezifikation /1/. Dies ist besonders deshalb kritisch, weil hier begangene Fehler, wie z. B. eine falsch interpretierte Zielstellung, eine ganze Entwicklung in Frage stellen können.

Ein Hauptproblem bei der Erarbeitung einer Anforderungsspezifikation bzw. eines Pflichtenheftes ist der Gegensatz zwischen einer ersten, verbalen und damit allgemein verständlichen Formulierung der Aufgabenstellung und deren formaler Darstellung.

Letztere ist unabdingbar, um die normalerweise in einer vom Auftraggeber erarbeiteten Problemstellung enthaltenen Widersprüche, Unvollständigkeiten und Unverträglichkeiten herausfinden und korrigieren zu können. Der Übergang von einer informalen zu einer formalen Darstellung ist aber mit einigen Problemen behaftet, die darin bestehen, daß

- der Übergang in der Regel auf einmal für das gesamte System erfolgt
- alle Einzelheiten bereits vor der Formalisierung festgelegt werden müssen
- ein Vergleich der entstandenen formalen Beschreibung mit der ursprünglichen verbalen Version nicht möglich ist.

Erschwerend wirken sich zudem die normalerweise vorhandenen Kommunikationsprobleme zwischen den im Umgang mit formalen Mitteln ungeübten Auftraggebern (meist der zukünftige Betreiber der Anlage) und den darin versierten Auftragnehmern (i. a. Softwareentwickler) aus.

Netze als Beschreibungsmittel

Werden Netze als Beschreibungsmittel eingesetzt, so ergeben sich gegenüber anderen Methoden folgende Vorteile:

① Netze lassen sowohl formale als auch informale Beschreibungen zu, das heißt, in einem ersten Schritt kann ohne Kenntnis eines Regelwerkes ein vorliegendes System grob modelliert werden. Dazu werden die Objekte des Systems und ihre Beziehungen untereinander spezifiziert, man beschreibt also die möglichen Objektzustände unter den verschiedenen Systembedingungen und deren Transformation durch Aktivitäten des Systems bzw. seiner Umgebung. An der Festlegung der relevanten Bedingungen und Aktivitäten des Systems kann sich sofort jeder Mitarbeiter beteiligen, was auch für eine mögliche und zunächst ebenfalls frei wählbare Beschriftung aller Netzelemente gilt. Wichtig ist, daß diese erste noch informale Darstellung bereits ein Netz ist und somit mit späteren formalisierten Entwurfsprodukten verglichen werden kann.

② Der Übergang von dieser ersten informalen Ebene zur formalen Darstellung muß bei Netzen nicht auf einmal und für das gesamte Netz erfolgen, sondern ist schrittweise und lokal möglich.

③ Modellpräzisierungen erfolgen nur dann, wenn es der Entwicklungsgang und Erkenntnisstand nötig machen und sind ebenfalls schrittweise und auf Teilsysteme begrenzt ausführbar.

Beispiel

Ausgangsbasis

Im folgenden werden diese Aussagen an einem Beispiel exemplarisch demonstriert. Bild 1 zeigt das Layout eines teilautomatisierten Fertigungsmaschinensystems /2/. Das System besteht aus vier Bearbeitungszentren (BAZ), die eine vollautomatische Fertigung von Werkstücken ausführen, einer Wasch- und Kühlstation (WAK) zur Finalbehandlung der bearbeiteten Teile sowie einem schienengeführten Transportroboter (STR), der den gesamten systeminternen Transport der Werkstücke übernimmt.

Die Schnittstelle nach außen bilden drei Spannplätze, bei denen die Rohteile auf Paletten aufgespannt und Fertigteile abgespannt werden. Wesentlich für den systeminternen Transport ist, daß bei den Bearbeitungszentren bzw. der Waschstation als Übergabeeinheiten Drehwechsler (DW) benutzt werden, die eine Beschickung der Ma-

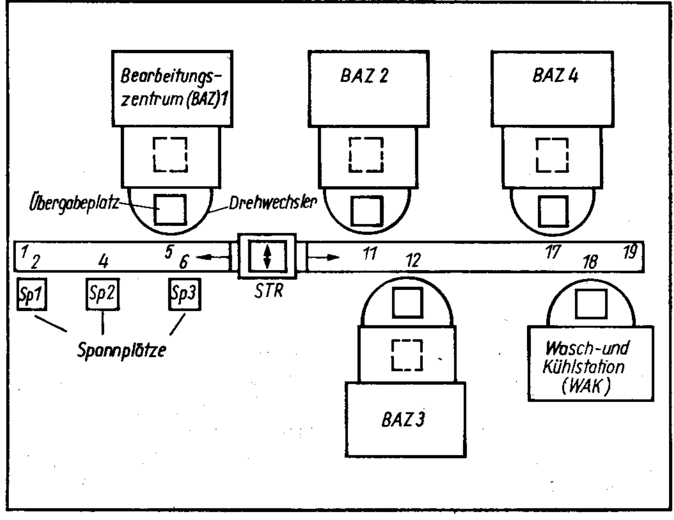


Bild 1 Layout des FMS

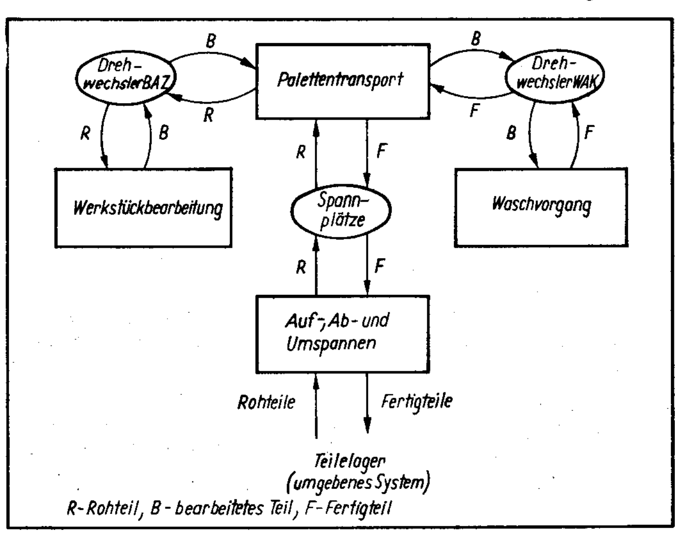


Bild 2 Informale Systemübersicht als Kanal-Instanzen-Netz

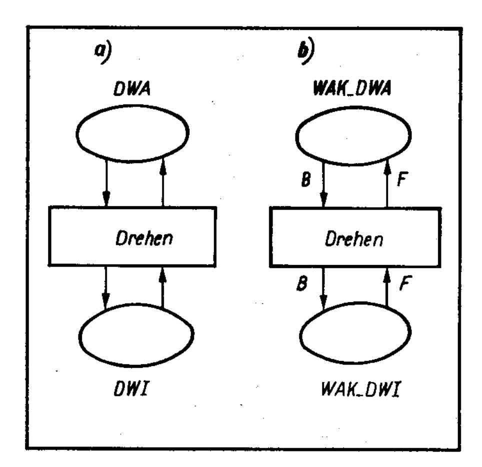


Bild 3 Verfeinerung des Drehwechslers der WAK

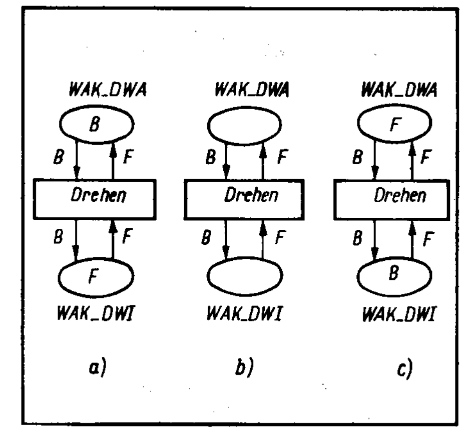
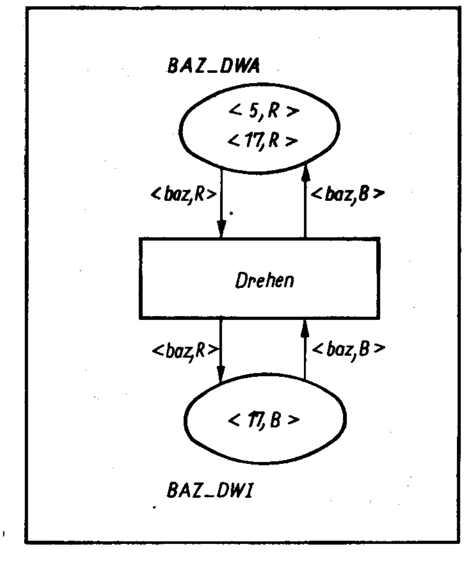


Bild 4 Objektverteilungen während des Drehens der WAK

Bild 5 Drehen des BAZ 4 (Position 17)



Dr.-Ing. Stephan Fensch absolvierte von 1973 bis 1977 ein Studium an der Technischen Hochschule Ilmenau und ein dreivierteljähriges Zusatzstudium am Moskauer Energetischen Institut. Sein Diplom erwarb er auf dem Fachgebiet Technische Kybernetik. Anschließend war er als wissenschaftlicher Assistent an der Technischen Universität Karl-Marx-Stadt beschäftigt; 1982 Promotion A zum Thema „Steuerung paralleler Prozesse". Von 1983 bis 1986 war er als Oberassistent im Wissenschaftsbereich Steuerungstechnik und Prozeßautomatisierung mit Forschungsarbeiten auf dem Gebiet der Anwendung der Netztheorie befaßt. Ab 1986 Einsatz im VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt; 1987 Delegierung in das Kombinat Trikotagen Karl-Marx-Stadt zur Realisierung eines komplexen Automatisierungsvorhabens.

Dipl.-Ing. Jürgen Lange studierte von 1980 bis 1984 an der damaligen Technischen Hochschule Karl-Marx-Stadt und diplomierte vorfristig auf dem Fachgebiet Technische Kybernetik zum Thema „Compiler für Netzsprache". Seit 1984 arbeitet er an der Sektion Automatisierungstechnik als wissenschaftlicher Assistent und beschäftigt sich mit der Theorie und Anwendung von Netzen, insbesondere mit der Entwicklung des Programmpaketes POSES.

schinen während der Teilebearbeitung ermöglichen.
Ausgehend von der Tatsache, daß die Bearbeitungszentren und die Waschstation komplette Teilsysteme mit separaten Steuerungen darstellen, besteht die Zielsetzung der Systemmodellierung in der Beschreibung der Organisation des Materialflusses.
In einem ersten Schritt ist es zunächst notwendig, unter dem Blickwinkel der angegebenen Zielstellung die Objekte des Systems, mit ihren Existenzbedingungen und Transformationen, zu beschreiben. Die Objekte des vorliegenden Systems sind offensichtlich die Werkstücke, die durch ihre Bearbeitungszustände *roh* (R), *bearbeitet* (B) und *fertig* (F) sowie eine Ortskoordinate charakterisiert werden können. Innerhalb des Systems befinden sie sich immer auf den zugehörigen Transportpaletten, so daß diese nicht explizit betrachtet werden müssen.
Folgende Systembedingungen gelten für ein Objekt:

– Es kann sich in den Zuständen *roh* oder *fertig* auf einem der Spannplätze befinden.
– Es kann sich in den Zuständen *roh* oder *bearbeitet* auf dem Drehwechsler eines der Bearbeitungszentren befinden.
– Es kann sich in den Zuständen *bearbeitet* oder *fertig* auf dem Drehwechsler der Waschstation befinden.
– Es kann sich in allen Zuständen auf dem Roboter befinden.

Die Transformation der Objektzustände erfolgt über die folgenden Aktivitäten:

– von *roh* auf *bearbeitet* bei konstantem Ort durch *Werkstückbearbeitung*
– von *bearbeitet* auf *fertig* bei konstantem Ort durch *Waschvorgang*
– Veränderung der Ortskoordinate bei konstantem Bearbeitungszustand durch *Palettentransport*.

Die Schnittstelle nach außen wird dabei durch die Aktivität *Spannen* realisiert.

Systemübersicht als Kanal-Instanzen-Netz

Aus diesen grundsätzlichen Überlegungen heraus läßt sich sofort eine noch informale Systemübersicht in Form eines Netzmodells angeben (Bild 2).
Die Systembedingungen sind dabei als Ovale (üblich sind auch Kreise), die Aktivitäten als Rechtecke dargestellt. Vom Standpunkt der Informationsverarbeitung läßt sich Bild 2 als ein sogenanntes Kanal-Instanzen-Netz /3/ auffassen. Dabei werden die Ovale als Kanäle interpretiert, die der Aufbewahrung oder Weitergabe von Objekten (Nachrichten) dienen, und die Rechtecke als Instanzen bezeichnet, die diese Objekte (Nachrichten) verarbeiten. Die Verbindung der Netzelemente wird über Pfeile realisiert, die entweder von Kanälen zu Instanzen oder von Instanzen zu Kanälen führen. Damit wird die Kausalstruktur *Vorbedingung – Aktivität – Nachbedingung* erzeugt, die das Grundgerüst aller Netzformen darstellt. Die Art der Objekte wird durch die Pfeilanschriften bezeichnet, das heißt, der Pfeil von *Spannplätze* zum *Paletten-Transport* mit der Anschrift *R* bedeutet, daß diese Instanz Zugriff auf den Kanal hat und ihm ein Objekt *Rohteil* entnehmen kann, wenn ein solches vorhanden ist.

Die Darstellung ist informal, weil es noch keine Regeln gibt, wann eine Instanz aktiv wird und welche Kanäle durch ihre Tätigkeit wie beeinflußt werden. Es ist aber schon ein wesentlicher Schritt in Richtung einer formalen Beschreibung getan, indem die Struktur des Systems mit seinen Funktionseinheiten fixiert ist und bei letzteren die *aktiven* Elemente (Instanzen) von den *passiven* (Kanälen) unterschieden werden können.
Um zu einer formalen Spezifikation zu kommen, muß die erste Grobdarstellung weiter untersetzt werden. Dies wird durch die Verfeinerung einzelner Netzelemente erreicht. Dabei gilt, daß eine Instanz ausschließlich durch ein mit Instanzen berandetes Teilnetz und ein Kanal nur durch ein mit Kanälen berandetes Teilnetz ersetzt werden kann.
Verfeinert werden nur die Elemente, die für die Systemspezifikation notwendig sind und für die die erforderlichen Details auch vorliegen.
So lassen sich die Drehwechsler durch einen äußeren und einen inneren Kanal sowie die Instanz *Drehen* sehr einfach und anschaulich modellieren (Bild 3a). Der technologische Ablauf z. B. an der Waschmaschine bestimmt nun, daß nur dann gedreht werden darf, wenn innen ein fertiges Teil und außen ein bearbeitetes Teil auf dem Drehwechsler liegen. Diese Forderung wird durch die im Bild 3b eingetragene Pfeilbeschriftung in das Modell übernommen. Die Instanz *Drehen* wird also genau dann aktiviert, wenn im Kanal *WAK_DWA* ein Objekt *B* und im Kanal *WAK_DWI* ein Objekt *F* vorhanden ist. Die Beschriftung der anderen beiden Pfeile gibt an, daß nach Ausführung der Instanz *Drehen* in Kanal *WAK_DWA* ein Objekt *F* und im Kanal *WAK_DWI* ein Objekt *B* liegt. Bild 4 demonstriert die Objektverteilungen auf dem Drehwechsler vor, während und nach Ausführung der Instanz *Drehen*.

Verfeinerung zum Prädikat/Transitionsnetz

Aus diesen Bildern wird ersichtlich, daß die hier angegebenen Instanzen aus allen Eingabekanälen Objekte benötigen. Sind diese in ausreichender Zahl vorhanden, so erhält die Instanz Konzession, was eine notwendige Bedingung für ihre Ausführung ist. Bei der Ausführung werden zunächst die entsprechenden Objekte aus allen Eingangskanälen entnommen, und ggf. wird eine der Instanz zugeordnete Prozedur gestartet (z. B. *Drehen*). Mit dem Ende der Ausführung werden abschließend in alle Ausgabekanäle auch wieder Objekte hineingelegt. Dieses Verhalten bezeichnet man als transitional,

das heißt, es gibt eine definierte Regel, die sogenannte Schaltregel, nach der die Konzessionserteilung und Ausführung einer Instanz genau festgelegt sind. Verhalten sich alle Instanzen des Netzes nach dieser Regel, so spricht man von einem Petri-Netz /3/, und die Instanzen werden als Transitionen bezeichnet.
Die Nachrichten (Objekte) werden in Petri-Netzen üblicherweise mit einem Punktsymbol in die Kanäle eingetragen und als Marke bezeichnet. Im einfachsten Fall gibt eine Nachricht nur Auskunft darüber, ob ein bestimmtes Objekt im Kanal vorhanden ist oder nicht (z. B. *Werkstück auf Drehwechsler).* Diese 0-1-Information über das Vorhandensein des Objekts im Kanal wird noch um eine Information über den Zustand des Objekts (*R, B* oder *F)* erweitert. Man sagt, das Objekt (Marke) trägt ein Attribut, das im konkreten Fall vom Typ Werkstück ist. Wie später noch gezeigt wird, kann dieses Attribut auch aus mehreren Teileigenschaften zusammengesetzt sein (mehrstelliges Attribut).
Solche Netze heißen Prädikat/Transitionsnetze (Pr/T-Netze), weil die Kanäle als Modellierung einer Eigenschaft der Objekte angesehen werden können, wenn diese als Marken im Kanal vorhanden sind. Aussagen dieser Art werden in der formalen Logik als Prädikate bezeichnet.
Analog zum Drehwechsler der Waschmaschine kann man nun den Kanal *Drehwechsler_BAZ* von Bild 2 verfeinern, indem man für jeden Drehwechsler ein entsprechendes Teilnetz angibt. Dies hätte allerdings zur Folge, daß auch die Instanz *Werkstückbearbeitung* in *Bearbeitung 1* bis *Bearbeitung 4* verfeinert werden müßte. Dabei vergrößert sich die Komplexität des Netzmodells unnötigerweise, da der technologische Ablauf der Werkstückübergabe an allen vier Bearbeitungszentren gleich ist.
Sollen die vier Drehwechsler mit nur einem Kanal modelliert werden, muß neben dem möglichen Objektzustand (R oder B) auch eine Spezifikation des Bearbeitungszentrums, auf dem sich das Objekt befindet, als weiteres Attribut eingeführt werden. Da alles transportorientiert betrachtet wird, bieten sich dafür die Positionen der Bearbeitungszentren entlang der STR-Schienenführung an. Liegen auf dem Kanal BAZ_DWA z. B. 2 Objekte der Form (5,R) und (17,B), so bedeutet das, daß sich außen auf dem Drehwechsler des Bearbeitungszentrums 1 (Position 5) ein Rohteil und auf dem des Bearbeitungszentrums 4 (Position 17) ein bearbeitetes Teil befindet. Die Drehwechsler der anderen beiden Bearbeitungszentren sind außen frei. Die formale Darstellung der Konzessionierungsbedingungen zeigt Bild 5 über die Pfeilbeschriftung, wobei für das neue Objektattribut die Variable *baz* eingeführt wird. So werden über den vom Kanal *BAZ_DWA* zur Instanz *Drehen* führenden und mit (baz,R) beschrifteten Pfeil 2stellige Objekte aus der Menge (⟨5,R⟩, ⟨11,R⟩, ⟨12,R⟩, ⟨17,R⟩) transportiert. Die Variable *baz* wird dabei gleichzeitig als Parameter für die Auswahl der Drehwechslerprozedur verwendet. Die im Bild 5 dargestellte Objektverteilung ermöglicht ein Drehen des Drehwechslers am Bearbeitungszentrum 4, das heißt, es kann die Prozedur *Drehen (17)* gestartet werden.
Für die Verfeinerung der Instanz *Palettentransport* von Bild 2 können zunächst einzelne Transportfahrten nach ihren Ziel-Quelle-Angaben unterschieden werden. Für den Normalbetrieb der Anlage gibt es dabei drei Grundtypen:

① von Spannplätzen zu Bearbeitungszentren (Spann_BAZ)

② von Bearbeitungszentren zur Waschstation (BAZ_WAK)

③ von der Waschstation zu Spannplätzen (WAK_Spann).

Bild 6 zeigt das Einpassen dieser drei Transporte als Transitionen in das Netzmodell. Dabei wird aber sofort ein Widerspruch zur Realität sichtbar, da das Modell offenbar zuläßt, daß die Transporte auch nebenläufig zueinander ausgeführt werden. Da aber nur ein Transportroboter zur Verfügung steht, ist diese Nebenläufigkeit im realen Prozeß nicht möglich. Der Roboter stellt also eine typische Systemressource dar, die natürlich im Modell berücksichtigt werden muß.
Bild 7 zeigt die korrekte Verfeinerung der Instanz *Palettentransport,* wobei der Transportroboter durch den Kanal *STR_FREI* modelliert wird. Dieser Kanal enthält genau eine Marke, die für die Konzessionserteilung an jede Transporttransition notwendig ist. Damit wird die geforderte Exklusivität der drei Teilprozesse gesichert. Um eine weitere Verfeinerungsstufe erreichen zu können, wird ein Ablaufalgorithmus aufgestellt, der für jede der drei Transportmöglichkeiten des Systems Gültigkeit hat:

1. Fahre leer von Roboter-Position zur Quellposition.

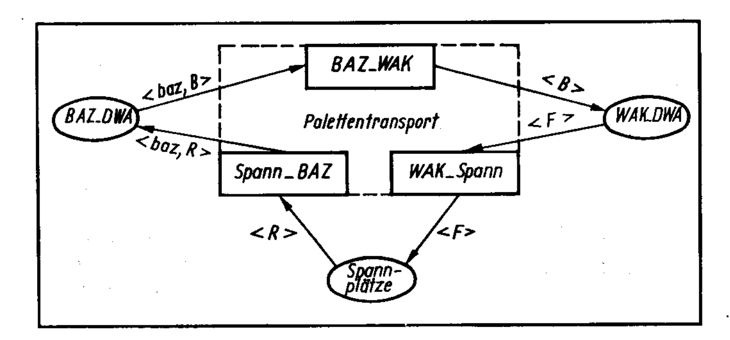


Bild 6 Verfeinerung der Instanz „Palettentransport"

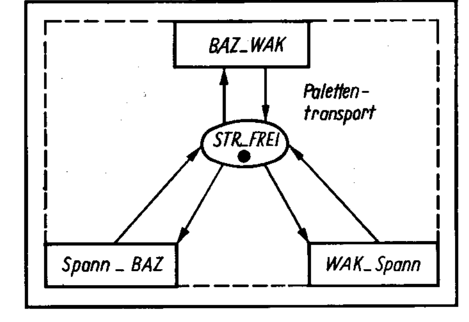


Bild 7 Modellierung des Transporters als Ressource

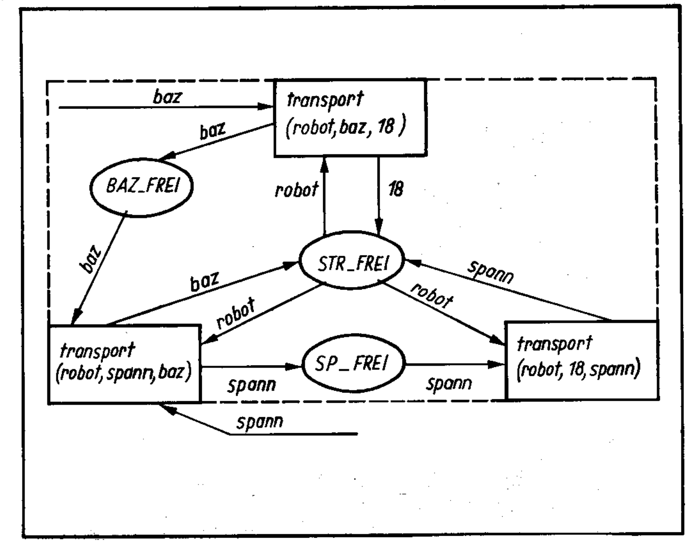


Bild 8 Weitere Verfeinerung der Transporte

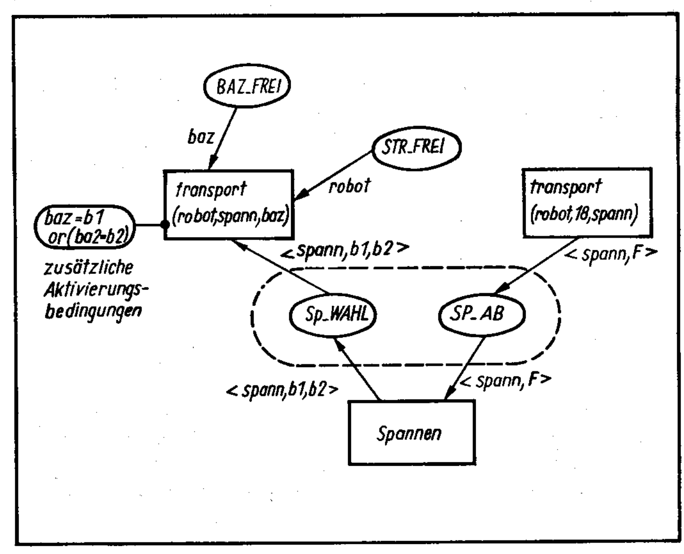


Bild 9 Verfeinerung des Spannens

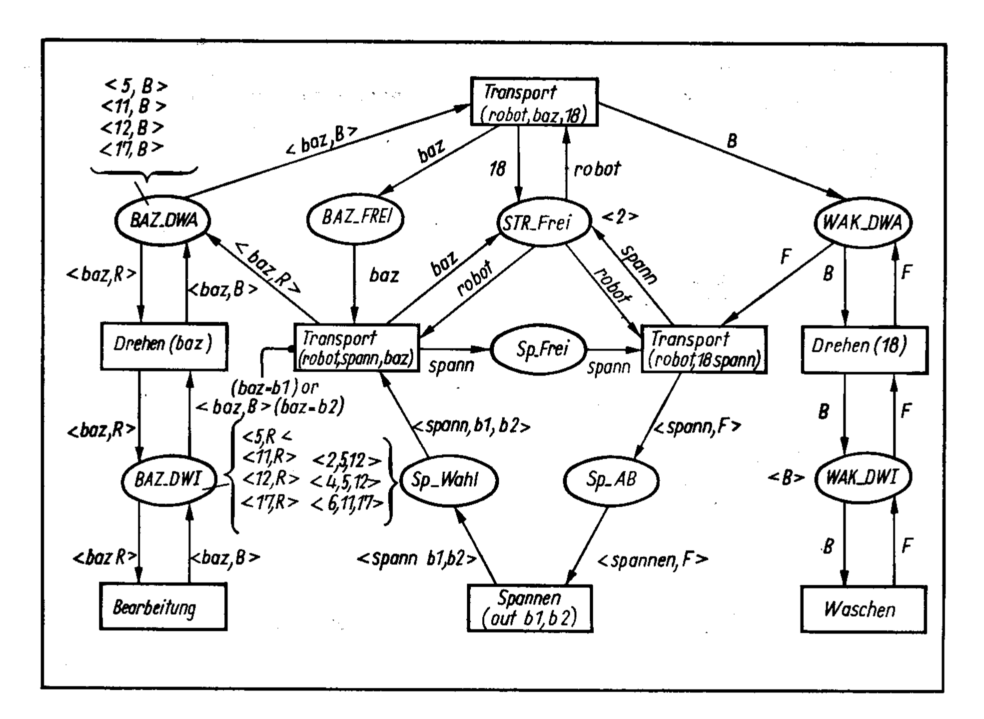


Bild 10 Prädikat-/Transitionsnetz des FMS

2. Übernehme Palette von rechts (wenn Position gerade) oder links (ungerade).
3. Fahre von Quellposition zur Zielposition.
4. Übergebe Palette nach rechts (wenn Position gerade) oder links (ungerade).

Damit läßt sich eine einheitliche Prozedur für die betrachteten Transportvarianten in der Form
transport (robot,quelle,ziel) angeben. Die drei Inputparameter sind alle vom Typ *position* (Wertebereich: 1...19, gemäß Bild 1) und repräsentieren die aktuelle Roboterposition sowie die Quell- und Zielposition des Transportauftrags. Mit dem Schalten einer Transporttransition wird die Prozedur *transport* gestartet. Dabei müssen die aktuellen Parameter durch die der Transition vorgelagerten Kanäle in Form von attributierten Marken zur Verfügung gestellt werden.
Für die aktuelle Roboterposition kann dazu der Kanal *STR_FREI* genutzt werden, wenn man die Marke um ein Attribut, das die aktuelle Roboterposition angibt, erweitert. Diese Marke wird beim Start einer Transportoperation aus dem Kanal entnommen und bei ihrem Ende mit dem Wert der erreichten Zielposition zurückgegeben. Wenn man berücksichtigt, daß für die WAK eine feste Position als Konstante benutzt werden kann (18), und daß sowohl von den Spannplätzen als auch den Bearbeitungszentren jeweils die Quellposition in den Marken mitgeliefert wird, fehlen für die Transitionen *Spann_BAZ* und *WAK_Spann* noch die Parameter der Zielposition. Ziel dieser Transporte können aber nur freie Bearbeitungszentren bzw. Spannplätze sein, so daß es genügt, die Nachricht über ihr Freiwerden in entsprechenden Kanälen abzulegen.
Bild 8 zeigt die Installation der beiden Kanäle *BAZ_FREI* und *SP_FREI* in die Instanz *Palettentransport* sowie den Einsatz der vereinheitlichten Transportprozedur. Wie die Variable *baz* sind die Variablen *robot* (Roboterposition) und *spann* (Position des ausgewählten Spannplatzes) vom Typ *position* und spezifizieren die konkrete Quell-Ziel-Angabe der Transportprozedur. Auf diesem Niveau des Modells lassen sich nun weitere Forderungen des Auftraggebers einbringen. So ist es z. B. häufig üblich, daß nicht alle Bearbeitungszentren auch alle Teile des zu fertigenden Sortiments bearbeiten, sondern daß mehrere Maschinengruppen gebildet werden, die unterschiedliche Bearbeitungen ausführen.
Im vorliegenden Fall sollen die 4 Bearbeitungszentren in 2 Gruppen zu je 2 Maschinen aufgeteilt sein. Beim Aufspannen der Teile gibt der Bediener dann die jeweils zugeordnete Maschinengruppe an. Im Modell wird die jeweilige Gruppe durch die beiden Positionsangaben der zugehörigen Bearbeitungszentren angegeben. Diese bilden zusammen mit der Spannplatzposition ein Objekt, welches von der Transition *Spannen* erzeugt wird. Um dieses Objekt zwischenspeichern zu können, muß der Kanal *Spannplätze* in zwei Kanäle verfeinert werden. Der Kanal *SP_WAHL* enthält die für eine konkrete Maschinengruppe ausgewählten Rohteile, während der Kanal *SP_AB* die abzuspannenden Fertigteile aufnimmt. Bild 9 zeigt diese Verfeinerung als Ausschnitt aus dem Gesamtnetz, wobei gegenüber Bild 2 auf das Attribut *Rohteil (R)* verzichtet wurde, da sich Objekte im Kanal *SP_WAHL* nur in diesem Zustand befinden können. Die Variablen *b1* und *b2* sind vom Typ *position* und symbolisieren die ausgewählte Maschinengruppe.
Diese Verfeinerung hat allerdings bzgl. der Ausführung des Transports *Spann_BAZ* eine Konsequenz, die ebenfalls dargestellt ist. Ein Transport kann nur stattfinden, wenn mindestens eines der beim Spannen ausgewählten Bearbeitungszentren einer Maschinengruppe auch beschickungsbereit ist. Das bedeutet, daß diese Transition nur dann Konzession erhält, wenn sich zusätzlich zu den auf den angegebenen Eingangskanälen notwendigen Objekten in der Menge der auf *BAZ_FREI* und *SP_WAHL* befindlichen Objekte mindestens ein Paar befindet, bei dem ein freies Bearbeitungszentrum (Variable *baz*) mit einer Anwahl (den Positionsangaben *b1* oder *b2*) übereinstimmt. Diese zusätzliche Bedingung kann bei den Pr/T-Netzen für jede Transition als Boolescher Ausdruck formuliert werden, hier:
(baz = b1) or (baz = b2).
Bild 10 zeigt das bisher erreichte Gesamtnetzmodell, bei dem die ursprüngliche Struktur von Bild 2 noch recht gut erkennbar ist. Die eingetragenen Anfangsmarkierungen, z. B. (2) für STR_FREI, entsprechen dem Zustand des Systems unmittelbar vor dem Start des bisher betrachteten regulären Betriebes.
Durch „Markenschieben" können jetzt spezielle Situationen oder Abläufe per Hand durchgespielt werden, um ein Gefühl für das modellierte System zu bekommen. Eine vollständige und komfortable Modellvalidierung ist allerdings nur mit Rechnerunterstützung möglich. Im zweiten Teil dieses Beitrags wird deshalb als Simulationswerkzeug für solche Modelle das auf Personalcomputer implementierte Simulationswerkzeug POSES vorgestellt.

wird fortgesetzt

Literatur

/1/ Winkler, P.: Anforderungsbeschreibung mit Netzmodellen. atp 28 (1986)1, S. 32–39
/2/ Flexible Manufacturing System from WMW. Firmenschrift WMW-Export-Import, S. 23
/3/ Reisig, W.: Systementwurf mit Netzen. Springer-Verlag 1985

✉ KONTAKT ☏

Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 5 61 34 36

Termine

16. Jahrestagung Grundlagen der Modellierung und Simulationstechnik

WER? Fachausschuß *Modellierung* der Wissenschaftlich-Technischen Gesellschaft für Meß- und Automatisierungstechnik in der Kammer der Technik
WANN? 8. bis 10. Dezember 1987
WO? Rostock
WAS?
- Simulationsmethoden und -techniken
- Modellbildung und Simulation in der Biotechnologie
- Einsatz von Modellen und Simulationsmethoden für CAD/CAM im Maschinenbau
- Methoden der Modellaggregation

WIE?
Teilnahmemeldungen schriftlich an:
Präsidium der KDT, Sekretariat der WGMA, PSF 1315, Berlin, 1086

Müller

Programmieren mit MACRO-SM

Teil II

Dr. Thomas Horn
Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden

1.7. Direktanweisung

Mit der *Direktanweisung* kann man Symbolen einen Wert zuweisen. Sie hat folgende Syntax:

name: *symbol* = *expr* ; *kommentar*

Das Namensfeld und das Kommentarfeld können auch entfallen. Dem Symbol *symbol* wird der Wert des Ausdruckes *expr* zugewiesen. Bei der ersten Wertzuweisung wird das Symbol in die UST eingetragen. Bei weiteren Wertzuweisungen wird nur der Wert des Symbols in der UST geändert.
Das Symbol kann mit == bzw. durch eine **.GLOBL**-Anweisung als global erklärt werden. Eine Bezugnahme auf globale nichtdefinierte Symbole ist nicht erlaubt. Bei der Berechnung des Ausdruckes sind einstufige Vorwärtsbezugnahmen auf Symbole zulässig. Das Verschieblichkeitsattribut des Symbols wird durch den Ausdruck bestimmt. In verschieblichen PA darf der Wert des Ausdrucks relativ oder absolut sein.

Beispiel:
```
ALPHA = BETA + 10
        ;VORWAERTSBEZUGNAHME
        ;AUF BETA
BETA = 100
        ;ZUWEISUNG EINES
        ;ABSOLUTEN WERTES
FELDN = FELD + N - 1
        ;ZUWEISUNG EINES
        ;RELATIVEN WERTES
```

1.8. Speicherplatzzuordnungszähler

Der Speicherplatzzuordnungszähler (SZZ) wird durch einen alleinstehenden Punkt (.) bezeichnet. In Verbindung mit Radix-50-Zeichen hat der Punkt die Bedeutung eines Alphazeichens oder eines Dezimalpunktes! Wird der SZZ im Operandenfeld verwendet, so entspricht sein Wert der Adresse des ersten Wortes des Maschinenbefehls. Bei Assembleranweisungen entspricht der Wert des SZZ der Adresse des aktuellen Wortes oder Bytes.

Beispiel:
```
M0: MOV .,R2
;DER SZZ ENTSPRICHT DEM SYMBOL
;M0
FELD: .WORD 20,ALPHA,.,20.
;SZZ ENTSPRICHT DER ADRESSE
;FELD + 4
```

Das Verschieblichkeitsattribut des SZZ wird durch den Charakter des PA bestimmt. Für nichtverschiebliche PA ist der SZZ absolut und für verschiebliche PA relativ. Am Anfang eines PA bzw. der Übersetzung wird dem SZZ immer der Wert 0 zugewiesen. Mit der Direktanweisung kann dem SZZ ein beliebiger Wert zugewiesen werden. Die Direktanweisung kann auch zum Reservieren von Speicherbereichen benutzt werden.

Beispiel:
```
. = 17500    ;DER SZZ WIRD AUF 17500
             ;GESETZT
. = . + 120  ;DER SZZ WIRD UM 120
             ;BYTES
             ;WEITER GERUECKT
```
Die letzte Schreibweise ist identisch mit der Anweisung
```
.BLKB 120    ;RESERVIERE 120 BYTES
```

2. Allgemeine Assembleranweisungen

In diesem Abschnitt werden die wesentlichen Eigenschaften der allgemeinen Assembleranweisungen beschrieben. Das Format der Assembleranweisungen entspricht den Festlegungen gemäß 1.1, wobei auch für Assembleranweisungen in der Regel ein Symbol im Namensfeld zulässig ist. Da die Verwendung von Symbolen im Namensfeld von Assembleranweisungen (mit Ausnahme der Definitionsanweisungen) nicht von grundsätzlicher Bedeutung ist, wird darauf im weiteren nicht besonders hingewiesen.

2.1. Anweisungen zur Auswahl von Assemblerfunktionen

Bestimmte Funktionen des Makroassemblers lassen sich mit der Anweisung
.ENABL *arg*
einschalten bzw. mit der Anweisung
.DSABL *arg*
ausschalten, wobei *arg* die Funktion des Makroassemblers entsprechend Tafel 4 spezifiziert.
Die Funktionsauswahl kann auch mit den Schaltern /**EN**:*arg* und /**DS**:*arg* in der Kommandozeile erfolgen. Die durch die Schalter beeinflußten Argumente haben gegenüber den Argumenten der Anweisungen den Vorrang.

2.2. Anweisungen zur Steuerung des Übersetzungsprotokolls

Das Übersetzungsprotokoll enthält standardgemäß für jede Anweisung ein Fehlerkennzeichenfeld, in dem maximal vier Fehler angezeigt werden, den Speicherplatzzuordnungszähler (SZZ), eine dezimale Nummer der eingelesenen Quellanweisung, eine Spalte für den Objektkode in der maximalen Länge von 3 Worten und eine Spalte für die vollständige Quellanweisung. Die Kopfzeilen eines jeden Blattes enthalten den Programm-

Tafel 4 Steuerfunktionen des Assemblers

Argument	Standard	Funktion
ABS	disable	Ausgabe des absoluten Maschinenkodes im OS-RW-Format. Konvertieren in das formatierte Binärformat (für den Absolutlader) des Dienstprogramms FLX.
AMA	disable	Die relative Adressierung (Modifikation 67 gemäß 5.3) wird bei der Assemblierung durch die absolute Adressierung ersetzt (Modifikation 37). Diese Funktion erleichtert den Programmtest.
CDR	disable	Die Zeichen ab Position 73 werden als Kommentar gewertet. Damit können die Positionen 73–80 bei einer Lochkarteneingabe als Problem-Folgefeld benutzt werden.
CRF	disable	Ausgabe einer Cross-Referenz-Liste. Bei **.ENABL CRF** wird eine Cross-Referenz-Liste erstellt.
FPT	disable	Gleitkommazahlen werden nicht gerundet (Truncation). Bei **.DSABL FPT** werden die Gleitkommazahlen gerundet.
GBL	enable	Alle undefinierten symbolischen Bezugnahmen (Referenzen) werden als global-undefinierte Bezugnahmen angenommen, für die zur Taskbuilderzeit eine Auflösung erwartet wird (Bezugnahme auf globale Symbole anderer Objektmodule). Bei **.DSABL GBL** führen alle offenen symbolischen Bezugnahmen zu undefinierten Symbolen, die mit einem U-Fehler im Assemblerprotokoll angezeigt werden.
LC	disable	Die Kleinbuchstaben des ASCII-Kodes werden bei der Eingabe des Quellfiles nicht in Großbuchstaben umgewandelt. Im GOST-Kode ist somit auch eine Verarbeitung von kyrillischen Schriftzeichen möglich. Bei **.DSABL LC** erfolgt eine Umkodierung der kleinen bzw. kyrillischen Schriftzeichen in große lateinische.
LSB	disable	Lokale Symbolblöcke werden gewöhnlich durch ein Symbol im Namensfeld oder durch eine **.PSECT**-Anweisung begrenzt. Bei **.ENABL LSB** wird ein lokaler Symbolblock aufgebaut, dessen Gültigkeit sich bis zur nächsten **.DSABL LSB**-, **.ENABL LSB**- oder **.PSECT**-Anweisung erstreckt.
PNC	disable	Ausgabe des Binärkodes. Bei **.DSABL PNC** wird die Ausgabe des Binärkodes (Maschinenkodes) unterdrückt.
REG	enable	Die Standard-Registersymboldefinitionen werden verwendet (**R0 = %0, R1 = %1, .., R5 = %5, SP = %6, PC = %7**). Bei **.DSABL REG** werden keine Standard-Registersymboldefinitionen vom Assembler durchgeführt.

```
PSECT - BEISPIELE          MACRO M28     Ø4-JUL-84 17:25  PAGE 1

     1                              .TITLE  PSECT - BEISPIELE
     2 ØØØØØØ                       .PSECT  PA1,ABS
     3 ØØØØØØ  Ø127ØØ   ST:         MOV     #1ØØ.,RØ
               ØØØ144
     4 ØØØØØ4  Ø127Ø1               MOV     #BER,R1
               ØØØØØØ'
     5 ØØØØ1Ø  ØØ5Ø21   1¤:         CLR     (R1)+
     6 ØØØØ12  ØØ53ØØ               DEC     RØ
     7 ØØØØ14  ØØ1375               BNE     1¤
     8 ØØØØØØ                       .PSECT
     9 ØØØØØØ           BER:        .BLKW   1ØØ.
    1Ø ØØØ31Ø           CØ:         .BLKW   4
    11 ØØØ32Ø           C1:         .BLKW   2
    12 ØØØØ16                       .PSECT  PA1
    13 ØØØØ16  Ø127ØØ               MOV     #CØ,RØ
               ØØØ31Ø'
    14 ØØØØ22  Ø127Ø1               MOV     #C1,R1
               ØØØ32Ø'
    15 ØØØØ26  ØØØ167               JMP     INPUT
               ØØØØØØ'
    16 ØØØØØØ                       .PSECT  EING
    17 ØØØØØØ  ØØ4767   INPUT:      JSR     PC,UPEIN
               ØØØØØØG
    18 ØØØØØ4  1Ø3ØØ2               BCC     1¤
    19 ØØØØØ6  ØØØ167               JMP     FEHLER
               ØØØØØØG
    2Ø ØØØØ12           1¤:
    21 ØØØ324                       .PSECT
    22 ØØØ324           PUFFER:     .BLKB   8Ø.
    23 ØØØ444           ZAEHL:      .BLKW   1
    24         ØØØØØØ               .END    ST

PSECT - BEISPIELE          MACRO M28     Ø4-JUL-84 17:25  PAGE 1-1
SYMBOL TABLE

BER      ØØØØØØR          FEHLER= ****** GX          ST       ØØØØØØ       ØØ2
CØ       ØØØ31ØR          INPUT   ØØØØØØR       ØØ3 UPEIN  = ****** GX
C1       ØØØ32ØR          PUFFER  ØØØ324R            ZAEHL    ØØØ444R

. ABS.   ØØØØØØ      ØØØ
         ØØØ446      ØØ1
PA1      ØØØØ32      ØØ2
EING     ØØØØ12      ØØ3
ERRORS DETECTED:  Ø

VIRTUAL MEMORY USED:  1Ø1 WORDS  ( 1 PAGES)
DYNAMIC MEMORY:  3Ø86 WORDS  ( 11 PAGES)
ELAPSED TIME:  ØØ:ØØ:Ø5
,PSECT/LI:TTM=PSECT
```

Bild 1 Beispiel eines Übersetzungsprotokolls mit Einteilung in Programmabschnitte

Tafel 5 Funktionen zur Protokollgestaltung

Argument	Standard	Funktion
SEQ	list	Die Quellzeilennummer wird gedruckt.
LOC	list	Der Speicherplatzzuordnungszähler (SZZ) wird gedruckt.
BIN	list	Der übersetzte Binärkode wird gedruckt.
BEX	list	Der erweiterte Binärkode wird gedruckt. Als erweiterter Binärkode wird der Kode bezeichnet, der in der Zeile der Quellanweisung nicht untergebracht werden kann und somit den Druck von Folgezeilen hervorruft. **.NLIST BIN** verbietet auch **BEX**.
SRC	list	Die Quellzeile wird gedruckt.
COM	list	Die Kommentare, die im Quelltext enthalten sind, werden gedruckt. **.NLIST SRC** verbietet auch **COM**.
MD	list	Makrodefinitionen und die Auflösungen der Wiederholungsblöcke werden gedruckt.
MC	list	Makrorufe und die **.IRP**/**.IRPC**-Anweisungen werden gedruckt.
ME	no list	Die Makroauflösung wird als Quell- und Binärkode gedruckt.
MEB	no list	Die binäre Makroauflösung wird gedruckt. Das Argument **MEB** ist eine Untermenge des Arguments **ME.**
CND	list	Es wird der Quelltext der nichtgenerierten Bedingungsblöcke und aller Anweisungen der bedingten Generierung gedruckt.
LD	no list	Es werden alle **.LIST**- und **.NLIST**-Anweisungen ohne Argumente gedruckt.
TOC	list	Die Subtiteltabelle wird gedruckt.
TTM	no list	Das Protokollformat wird an das Terminalformat (72 Zeichen pro Zeile) angepaßt. Die Worte der binären Auflösung werden untereinander gedruckt. Bei **.NLIST TTM** wird das Paralleldruckerformat (132 Zeichen pro Zeile) ausgegeben.
SYM	list	Die Symboltabelle wird gedruckt.

titel, den Identifikator des Makroassemblers, Datum und Uhrzeit und eine Seitennummer (Bild 1)

Für die Steuerung des Übersetzungsprotokolls können folgende Anweisungen verwendet werden:

.TITLE – Programmtitel/Modulname
.IDENT – Identifikation des Moduls
.SBTTL – Untertitel des Quellprogramms
.PAGE – Beginn einer neuen Seite
.LIST – Listen
.NLIST – Nicht listen.

Die **.TITLE**-Anweisung dient zur Bezeichnung des Objektmoduls und des Übersetzungsprotokolls mit dem Programmtitel. Sie hat folgende Syntax:

.TITLE *string*

wobei *string* den Programmtitel festlegt. Die ersten 6 Zeichen müssen im Radix-5Ø-Kode zulässig sein und werden als Modulname zur Bezeichnung des Moduls verwendet. Die restlichen Zeichen dürfen beliebige ASCII-Zeichen sein. Die ersten 31 Zeichen des Programmtitels werden in die Kopfzeile eingefügt und auf jedem Blattanfang gedruckt. Wird die **.TITLE**-Anweisung nicht spezifiziert, erhält der Modul den Namen .MAIN.

Die **.IDENT**-Anweisung wird zur zusätzlichen Bezeichnung des Moduls mit einem Versionsidentifikator verwendet. Die Anweisung hat folgende Syntax:

.IDENT */string/*

wobei *string* der Versionsidentifikator aus maximal 6 Radix-5Ø-Zeichen ist, der in Zeichenkettenbegrenzer eingeschlossen werden muß. Als Zeichenkettenbegrenzer können beliebige Sonderzeichen verwendet werden.

Die **.SBTTL**-Anweisung wird zur zusätzlichen Beschriftung des Assemblerprotokolls mit Untertiteln benutzt. Der jeweils letzte Untertitel wird bei Beginn einer neuen Seite als zweite Kopfzeile gedruckt. Vor der ersten Seite des Übersetzungsprotokolls wird eine Zusammenfassung aller Untertitel mit Angabe der Seiten- und Quellzeilennummer als Inhaltsverzeichnis (TOC – Table of contents) gedruckt. Die Anweisung hat folgende Syntax:

.SBTTL *string*

wobei *string* der Untertitel ist. Die Numerierung der Seiten erfolgt durch zwei Nummern: x–y, wobei

x eine logische Seitennummer und
y eine Blattnummer innerhalb der logischen Seite

darstellt.

Die Blattnummer wird ab Null beginnend gezählt, wobei eine Null nicht gedruckt wird. Die logische Seitennummer wird ab Eins beginnend gezählt. Eine Erhöhung der logischen Seitennummer erfolgt durch eine **.PAGE**-Anweisung oder durch Verwendung des Formularsteuerzeichens (FF) als Zeilenbegrenzer. Die **.PAGE**-Anweisung hat keinen Parameter. Jede neue logische Seite wird auf ein neues Blatt gedruckt, wobei die Blattnummer auf den Anfangswert rückgesetzt wird.

Mit der **.LIST**- bzw. **.NLIST**-Anweisung ohne Argument wird die Protokollierung der Quellzeilen (einschl. Objektkode) erlaubt oder unterdrückt. Die Protokollierung ist erlaubt, wenn ein sogenannter Zähler für das Protokollierungsniveau einen positiven Wert hat. Bei einem negativen Wert erfolgt keine Protokollierung. Der Anfangswert des Zählers ist Null. Durch eine **.LIST**-Anweisung wird der Zähler inkrementiert und durch eine **.NLIST**-Anweisung dekrementiert.

Mit einer **.LIST**- bzw. **.NLIST**-Anweisung mit Argument

.LIST *arg[,arg[,...]]*
.NLIST *arg[,arg[,...]]*

wird die Gestaltung des Protokolls beeinflußt. Der Zähler für das Protokollierungsniveau wird nicht verändert. Die zulässigen Argumente mit ihren Standardwerten sind in Tafel 5 zusammengefaßt.

Die Protokollgestaltung kann auch durch die Schalter **/LI:***arg* und **/NL:***arg* in der Kommandozeile beeinflußt werden. Die durch die Schalter beeinflußten Argumente haben gegenüber den Argumenten der **.LIST**- und **.NLIST**-Anweisung den Vorrang.

2.3. Definitionsanweisungen

Mittels Definitionsanweisungen können Speicherbereiche reserviert oder Zeichenketten und Konstanten in den Objektkode eingetragen werden. Symbole im Namensfeld der Definitionsanweisungen sind erlaubt. Den Symbolen wird der aktuelle Wert des SZZ, d. h. die Anfangsadresse des Bereiches bzw. die Adresse des ersten Bytes der Zeichenkette bzw. der Konstanten, zugeordnet.

2.3.1. Definition von Speicherbereichen

Speicherbereiche können mit den folgenden Anweisungen definiert werden:

.BLKB *expr*
.BLKW *expr*

Die Bereiche werden durch die **.BLKB**-Anweisung in Bytes und durch die **.BLKW**-Anweisung in Worten reserviert, wobei der Ausdruck *expr* die Anzahl der Bytes bzw. Worte festlegt. Der Ausdruck muß absolut sein und darf keine Vorwärtsbezugnahmen oder Bezugnahmen auf externe Symbole enthalten. Die **.BLKW**-Anweisung muß auf einer geradzahligen Adresse beginnen.

Beispiele:
FELD: .BLKW 100
STACK: .BLKB N+OP+2

2.3.3. Definition von Zeichenketten

Die Definition von Zeichenketten im ASCII-Kode bzw. im Radix-50-Kode erfolgt mit den Anweisungen

.ASCII /string/.../string/
.ASCIZ /string/.../string/
.RAD50 /string/.../string/

Mit der **.ASCII**-Anweisung wird ein Speicherbereich reserviert, in den die Kodes der Zeichen der Zeichenkette *string* eingetragen werden, wobei die Zeichen / / die Begrenzer der Zeichenkette darstellen. Die Größe des reservierten Speicherbereiches wird durch die Anzahl der Zeichen festgelegt. Als Begrenzer können beliebige druckbare Zeichen außer <, = und ; benutzt werden. Jedoch darf ein Zeichen, welches innerhalb der Zeichenkette verwendet wird, kein Begrenzer sein. Die Zeichenkette darf in mehrere Teilzeichenketten unterteilt werden. Nichtdruckbare Zeichen dürfen in der Zeichenkette *string* nicht verwendet werden. Sie können durch einen Ausdruck mit dem entsprechenden Wert, der in spitze Klammern eingeschlossen werden muß, definiert werden. Durch einen Ausdruck können beliebige Zeichen definiert werden, jedoch darf der Ausdruck den Wert 377(8) nicht überschreiten. Ausdrücke und Zeichenketten dürfen in beliebiger Reihenfolge definiert werden.

Beispiele:
TEXT: .ASCII +BEGINN
DES PROGRAMMLAUFS+
TEXT1: .ASCII ⟨15⟩⟨12⟩
#LESEFEHLER VON DK0:/DK1 :#
LF=12
CR=15
TEXT2: .ASCII ⟨LF⟩/∗∗∗FEHLER/
⟨CR⟩⟨LF⟩? KODE:?

Die **.ASCIZ**-Anweisung entspricht der **.ASCII**-Anweisung, mit dem Unterschied, daß nach dem letzten Byte der Zeichenkette noch ein zusätzliches Byte mit dem Wert 0 als Endekennzeichen definiert wird.
Die **.RAD50**-Anweisung dient der Definition von Zeichenketten im Radix-50-Kode. Auf Grund der Einschränkung des Zeichenvorrates besteht die Möglichkeit, nach der Radix-50-Formel drei Zeichen in zwei Bytes zu definieren. Im Operandenfeld können analog der **.ASCII**-Anweisung Zeichenketten und Ausdrücke in spitzen Klammern verwendet werden. Ein Ausdruck darf den Wert von 47(8) nicht überschreiten. Eine Zeichenkette wird immer in Dreiergruppen unterteilt, die in den Radix-50-Kode umgewandelt und in ein Wort eingetragen werden. Die letzte unvollständige Dreiergruppe wird mit Leerzeichen aufgefüllt.

Beispiele:
A: .RAD50 /ABCDE/
B: .RAD50 ⟨1⟩⟨2⟩⟨3⟩⟨4⟩⟨5⟩
Die Zeichenketten **A** und **B** sind identisch und belegen zwei Worte. Die letzte Dreiergruppe wird durch ein Leerzeichen ergänzt.

2.3.3. Definition von Konstanten

Für alle hardwaremäßig unterstützten Datenformate existiert jeweils eine Definitionsanweisung:

.BYTE *expr[,expr[,...]]*
.WORD *expr[,expr[,...]]*
.FLT2 *arg[,arg[,...]]*
.FLT4 *arg[,arg[,...]]*

Mit der **.BYTE**-Anweisung werden ganze und natürliche Zahlen im Byteformat definiert. Die Anzahl der definierten Bytes entspricht der Anzahl der Ausdrücke. Die Ausdrucksberechnung erfolgt mit 16-Bit-Werten, und anschließend wird das Ergebnis auf 8 Bit verkürzt. Ist der Wert nicht in 8 Bit darstellbar, so wird ein T-Fehler angezeigt. Globale Bezugnahmen sind zulässig.

Beispiele:
C1: .BYTE 20,16.,^B10000,2∗8.,4+
⟨2∗6⟩, '0–40
C2: .BYTE ^0377,255,–1
C3: .BYTE W0
.GLOBL W0
Im Bereich **C1** werden 6 Bytes mit dem Wert 20(8) und im Bereich **C2** 3 Bytes mit dem Wert 377(8) definiert. Unter **C3** wird 1 Byte definiert, dessen Wert erst durch den Taskbuilder eingetragen werden kann.

Mit der **.WORD**-Anweisung werden analog der **.BYTE**-Anweisung ganze und natürliche Zahlen im 16-Bit-Format definiert. Eine **.WORD**-Anweisung muß immer auf einer geradzahligen Adresse beginnen. Die Ausdrücke dürfen relative und absolute Werte ergeben sowie globale Bezugnahmen enthalten. Es ist zu beachten, daß alle Anweisungen ohne Operationskode bzw. ohne gültigen Operationskode der **.WORD**-Anweisung gleichgesetzt sind.

Beispiele:
V0: .WORD FELD+100., ALPHA
;DEFINITION VON 2 ADRESS-
KONSTANTEN,
.WORD B101 110000111, "AB
;DEFINITION VON 2 NUMERISCHEN
;KONSTANTEN
Diese Anweisungen lassen sich auch wie folgt schreiben:

V0: FELD+100., ALPHA
^B101 110000111,"AB

Unter Anwendung der letzten Regel lassen sich z. B. neue Befehlsmnemoniks für adressenlose Befehle einführen:
CLNZ=254; BEFEHL ZUM LOESCHEN
;VON N- UND Z-BIT
SVC=104410; TRAP 10
...
...
CLNZ ;GENERIERUNG DER
;KONSTANTEN 000254(8)
SVC ;GENERIERUNG DER
;KONSTANTEN 104410(8)

Gleitkommazahlen werden mit der **.FLT2**-Anweisung im 2-Wort-Format und mit der **.FLT4**-Anweisung im 4-Wort-Format definiert, wobei jedes Argument zur Generierung einer Gleitkommazahl führt. Die Argumente werden nach den üblichen Regeln mit oder ohne Dezimalpunkt und mit oder ohne Exponent geschrieben. Ausdrücke sind als Argument unzulässig.

Beispiele:
FELD: .FLT2 2,+2,2.,2.0,2E0,
2.0E+0,20.E–1
K0: .FLT4 .31416E1,
1.3803E–16,–273.16,
5.61E26
Die erste Anweisung zeigt 7 verschiedene Schreibweisen für die Gleitkommakonstante +2, die noch um zahlreiche Varianten erweitert werden könnte.

2.4. Anweisungen zur Steuerung des Speicherplatzzuordnungszählers

Der Speicherplatzzuordnungszähler (SZZ) wird bei der Assemblierung von Befehlen und Speicherbereichsdefinitionen automatisch um die Länge des Befehls bzw. des Speicherbereiches inkrementiert. Mit der Direktanweisung (siehe 1.7.) kann der SZZ auf einen bestimmten Wert gesetzt bzw. um einen bestimmten Wert weitergezählt werden. Außerdem kann der SZZ durch die

.EVEN-Anweisung und
.ODD-Anweisung

beeinflußt werden.
Die **.EVEN**-Anweisung setzt den SZZ immer auf eine geradzahlige Adresse (Wortgrenze), indem zu einem ungeradzahligen Wert des SZZ eine Eins addiert wird. Ein geradzahliger Wert wird nicht verändert.
Die **.ODD**-Anweisung setzt den SZZ immer auf eine ungeradzahlige Adresse, indem zu einem geradzahligen Wert des SZZ eine Eins addiert wird. Ein ungeradzahliger Wert wird nicht verändert.

Beispiele:
TEXT7: .ASCII /EINGABE VON T0:/
.EVEN
;EINSTELLEN DER
;WORTGRENZE
DEZ: .WORD 100.,10.,1
;TABELLE IM WORT-
;FORMAT

...

```
         .ODD ;SZZ UNGERAD-
         ZAHLIG
SATZ:    .BYTE 2*ANZ
         ;SATZLAENGE IM BYTE-
         ;FORMAT
BER:     .BLKW ANZ
         ;DATENSATZ AUF WORT-
         ;GRENZE
```

2.5. Generelle Assembleranweisungen

In diesem Abschnitt werden einige Assembleranweisungen von allgemeiner Anwendung, wie

.END – Ende des Quellprogramms
.GLOBL – Definition globaler Symbole
.RADIX – Festlegung der Basis des Zahlensystems

beschrieben.
Die **.END**-Anweisung kennzeichnet das Ende eines Quellmoduls. Sie hat folgende Syntax:

.END *[expr]*

Der Ausdruck *expr* gibt die Startadresse des Programmsystems an. Dadurch wird der automatische Programmstart mit dem RUN-Kommando des Betriebssystems ermöglicht. Wenn mehrere Moduln zu einer Task verbunden werden, so ist die Angabe der Startadresse bei dem Modul notwendig, der als erster gestartet wird. Ist in keinem der Moduln eine Startadresse spezifiziert, so kann der Programmstart nur über das Testhilfesystem (ODT) erfolgen, wenn es in die Task eingebunden wurde.

Die **.GLOBL**-Anweisung wird zur Spezifikation globaler Symbole benutzt, die nicht mittels ":: " bzw. "==" als globale Symbole definiert wurden. Globale Symbole können globale Definitionen und Bezugnahmen realisieren.

Wird mit **.DSABL GBL** die automatische Definition aller undefinierten Bezugnahmen als globale Referenzen unterdrückt, können globale Referenzen über die **.GLOBL**-Anweisung explizit definiert werden. Die **.GLOBL**-Anweisung hat folgende Syntax:

.GLOBL *symbol[,symbol[,...]]*

wobei
symbol das als global zu spezifizierende Symbol und
, ein zulässiges Trennzeichen (Komma, Tabulator und/oder Leerzeichen) darstellen.
Die **.RADIX**-Anweisung legt die Basis des Zahlensystems für das Programm fest. Diese Basis wird zugrunde gelegt, wenn im Programm Zahlen ohne Angabe einer temporären Basis (siehe 1.5.) verwendet werden. Die Anweisung hat folgende Syntax:

.RADIX *[n]*

wobei *n* die Basis des Zahlensystems ist, die bis zur nächsten **.RADIX**-Anweisung gilt. Die Dezimalzahl *n* darf die Werte **2, 8** und **10** annehmen. Wenn *n* ausgelassen wird, so wird die Basis **8** angenommen, die ebenfalls gilt, wenn im Programm keine **.RADIX**-Anweisung enthalten ist.

Tafel 6 Argumente der **.PSECT***-Anweisung*

Argument	Standard	Funktion
RO oder **RW**	**RW**	PA ist nur lesbar (**RO**) oder auch beschreibbar (**RW**).
I oder **D**	**I**	PA mit Befehlen und Daten (**I**) oder PA mit Daten (**D**).
GBL oder **LCL**	**LCL**	Globaler PA einer Überlagerungsstruktur (**GBL**) oder PA mit lokaler Gültigkeit in einem Überlagerungssegment.
ABS oder **REL**	**REL**	Absoluter PA (**ABS**) oder relativer PA (**REL**).
OVR oder **CON**	**CON**	Statische Überlagerung von PAs mit gleichem Namen (**OVR**) oder sequentielle Anordnung von PAs mit gleichem Namen (**CON**).

Tafel 7 Bedingungen der **.IF/.IIF***-Anweisung*

Bedingung positiv	negativ	Argument(e)	Erläuterung
EQ oder Z	**NE** oder NZ	expr	Der Ausdruck wird auf 0 (bzw. ungleich 0) getestet.
GT oder **G**	**LE**	expr	Der Ausdruck wird auf größer als 0 (bzw. kleiner oder gleich 0) getestet.
LT oder **L**	**GE**	expr	Der Ausdruck wird auf kleiner als 0 (bzw. größer oder gleich 0) getestet.
DF	**NDF**	symbol bzw. log. Ausdr.	Es wird getestet, ob das Symbol oder die Symbole entsprechend der logischen Verknüpfung definiert (oder nicht definiert) sind.
B	**NB**	Makroparameter	Es wird geprüft, ob der Makroparameter ausgelassen (blank) oder nicht ausgelassen (no blank) wurde.
IDN	**DIF**	zwei Makroparameter	Es wird geprüft, ob die zwei Makroparameter identisch (oder verschieden) sind.

2.6. Einleitung von Programmabschnitten

Ein Programmabschnitt wird durch die Anweisung

.PSECT *[name][,arg[,arg[,...]]]*

eingeleitet, wobei

name – den Namen des Programmabschnittes festgelegt,
, – ein zulässiges Trennzeichen (Komma, Tabulator und/oder Leerzeichen) und
arg – ein Argument zur Spezifikation von Attributen zur Auswahl von bestimmten Eigenschaften des Programmabschnittes gemäß Tafel 6 bedeuten.

Der Makroassembler läßt in einem Quellmodul 256 Programmabschnitte (PA) zu:

– einen absoluten (nichtverschieblichen) PA, mit dem Namen **. ABS.** (Standard),
– einen relativen (verschieblichen) unbenannten PA (Standard),
– 254 benannte PA.

Für jeden PA wird während der Übersetzung ein SZZ geführt, der ab Null beginnend in relativen oder absoluten Adressen, dem Attribut des PA entsprechend, geführt wird. Somit wird auch den Symbolen im Namensfeld der Anweisungen das Attribut absolut oder relativ zugeordnet. Ein PA muß im Quellprogramm nicht zusammenhängend hintereinander geschrieben werden, sondern kann durch Verwendung der gleichen **.PSECT**-Anweisung weiter unten im Programm fortgesetzt werden. Die Übersetzung wird dann mit dem letzten Stand des SZZ des entsprechenden PA weitergeführt.
Wenn ein Programm ohne **.PSECT**-Anweisung beginnt, so wird ein unbenannter verschieblicher PA aufgebaut, der später auch mit einer **.PSECT**-Anweisung ohne Namen fortgesetzt werden kann.

3. Die bedingte Assemblierung

Die bedingte Assemblierung gestattet die Generierung von verschiedenen Applikationsprogrammen, die an den jeweiligen Anwendungsfall angepaßt sind, aus einem Quellprogramm. Eine grundlegende Bedeutung haben diese sprachlichen Ausdrucksmittel für die Makrotechnik, da sie während der Makrogenerierung die flexible Anpassung der Makros an die verschiedenen Anwendungsfälle gestatten.

3.1. Die Anweisungen .IF und .ENDC

Ein bedingter Anweisungsblock wird mit einer **.IF**-Anweisung eingeleitet und mit einer **.ENDC**-Anweisung abgeschlossen. Bedingte Anweisungsblöcke dürfen ineinander geschachtelt werden. Ein bedingter Anweisungsblock hat folgenden Aufbau:

.IF *cond,arg[,arg]*
...
...
Folge von Anweisungen
...
...
.ENDC

wobei die Parameter folgende Bedeutung haben:

cond – spezifiziert entsprechend Tafel 7 eine Bedingung, auf deren Erfüllung das Argument bzw. die Argumente getestet werden. Wenn sich der Wahrheitswert "true" ergibt, so werden die nachfolgenden Anweisungen bis zur **.ENDC**-Anweisung mit übersetzt. Ansonsten werden diese Anweisungen bei der Übersetzung übergangen.
, – zulässiges Trennzeichen (Komma, Tabulator und/oder Leerzeichen)
arg – spezifiziert entsprechend Tafel 7 ein oder zwei Argumente, die Ausdrücke, Symbole oder formale Parameter einer Makrodefinition sein können.

wird fortgesetzt

Logische Werte im Standard-BASIC

Prof. Dr. Horst Völz
Berlin

BASIC gestattet zwar den Umgang mit logischen Werten und Funktionen, verfügt jedoch über keine spezifischen Größen. Hieraus ergeben sich verschiedene Konsequenzen und unerwartete Möglichkeiten. Sie werden im Beitrag untersucht. Ausgegangen wird dabei von dem gemeinsamen BASIC-Kern der KC 85/1 bis KC 85/3 und KC 87. Auf Besonderheiten in anderen BASIC-Dialekten wird verwiesen.

1. Wahrheitswerte

Wir haben in der Umgangssprache Aussagen, die *wahr* oder *falsch* sein können. Die Aussage: *4 ist eine Primzahl* ist falsch. Die Aussage: *4 ist gerade* ist dagegen wahr. In mehreren Rechnersprachen existieren nun besondere Variablen – Boolesche –, die nur zwei Werte, z. B.: 0/1, annehmen können. In BASIC werden derartige Aussagen über spezielle Zahlenwerte realisiert.
Zu den Booleschen Variablen führt immer ein Vergleich, der durch die Anweisung IF eingeleitet wird. Für den Vergleich werden die Operatoren:

=, <, >, < =, > = und < >

verwendet. Diese Operatoren sind anwendbar auf:
1. Numerische Variablen und Zahlenwerte, also z. B.:

A > B, A < 7, 5 = 9

2. Stringvariablen, Zeichenketten und Zeichen, also z. B.:

A$ < BF$, A$>"DE", STR$ (45) < > A$, "a" > "+"

Derartige Vergleiche sind wahr oder falsch. Hiervon kann man sich leicht mit PRINT im Direktmode überzeugen. So liefert:

PRINT 3 > 5
0

Dies bedeutet, daß der Zahlenwert 0 für falsch steht. Dagegen bringt:

PRINT 3 < 5
−1

Dieser Wert steht also für wahr. Nur in einigen speziellen Dialekten wird +1 für wahr verwendet.
Umgekehrt kann man nun prüfen, welche Zahlenwerte unter IF als wahr bzw. falsch interpretiert werden. Dies kann u. a. mit dem folgenden Programm erfolgen:

```
10 FOR X = -2 TO 2 STEp .25
20 PRINT X,: IF X THEN PRINT "wahr": ELSE
PRINT "falsch"
30 NEXT
```

Hieraus folgt, daßt alle Zahlen ungleich Null als wahr und nur Null als falsch interpretiert werden. Die Ergebnisse zeigt Bild 1.

2. Variablen als Wahrheitswerte

Wenn nun schon alle Zahlenwerte wie Wahrheitswerte behandelt werden, müßte man die Wahrheitswerte auch Variablen zuweisen können. Dabei entsteht eine etwas seltsam anmutende Schreibweise, wie:

LET A = A$ > B$, A = 3 > 4 oder gar A = B = C

Von dem Ergebnis derartiger Gleichungen kann man sich schnell durch PRINT A überzeugen. Es besitzt dann immer den Wert 0 oder −1.
Folglich behandelt der Interpreter die Schreibweise:

A = B = C

wie folgt: Er wertet zunächst das zweite Gleichheitszeichen im Sinne eines Vergleichs aus, und er weist dann diesen Wert mit dem ersten Gleichheitszeichen der Variablen A zu.
Es ist auffällig, daß diese Schreibweise analog zur Anweisung IF wirkt. Daraus ergibt sich die Möglichkeit, Wahrheitswerte zur wiederholten Nutzung in einer Variablen abzulegen. So läßt sich insbesondere bei komplizierten Vergleichen Speicher- und Rechenzeit einsparen. In diesem Sinne wurde bereits die Variable X im obigen Programm in Zeile 20 genutzt.

3. Logische Funktionen

Im Standard-BASIC existieren drei logische Funktionen:

NOT A, A OR B und A AND B

Die NOT-Funktion kehrt den Wahrheitswert um:

NOT (wahr) → falsch bzw. NOT (falsch) → wahr

Die OR- bzw. AND-Funktion sind am besten über eine Tabelle darstellbar:

A	B	AND	OR
w	w	w	w
w	f	f	w
f	w	f	w
f	f	f	f

Darin bedeuten w = wahr und f = falsch. Hierfür können auch −1 und 0 gesetzt werden. Es gelten verbal also die Aussagen:

A AND B ist nur wahr, wenn A und B gemeinsam wahr sind.
A OR B ist nur falsch, wenn A und B gemeinsam falsch sind.

Damit weicht das OR vom üblichen Sprachgebrauch des „oder" ab.

4. Zahlen bezüglich der logischen Funktionen

Nachdem sich zeigte, daß beliebige Zahlenwerte genauso wie die logischen Werte 0 und −1 wirken, ist es interessant zu prüfen, wie sich die logischen Funktionen bezüglich der Zahlenwerte verhalten. Auch dies erfolgt wieder bequem im Direktmode:

A = 137: B = 56: PRINT NOT A, NOT B
−138 −57
oder:
C = −3: D = −1200: PRINT NOT A, NOT B
2 1199

Mit NOT wird also zunächst das Vorzeichen gewechselt. Die entstehende Zahl wird dann noch um 1 verkleinert. Dies liefert zwar den Spezialfall bezüglich der Wahrheitswerte 0 und −1, ist aber eben viel allgemeiner.
Bezüglich AND und OR können wir uns nun mit

PRINT A AND B, A OR B
8 185

überzeugen. Hier ist die Analyse des Geschehens jedoch komplexer. Hierzu kann das Programm ANDOR dienen. Es wandelt jede vorhandene Zahl in drei Schritten um. Zunächst wird eine ganze Zahl gebildet, dann wird das Vorzeichen abgetrennt. Der Rest wird wie eine binäre 15-Bit-Zahl behandelt

-2	wahr
-1.75	wahr
-1.5	wahr
-1.25	wahr
-1	wahr
-.75	wahr
-.5	wahr
-.25	wahr
0	falsch
.25	wahr
.5	wahr
.75	wahr
1	wahr
1.25	wahr
1.5	wahr
1.75	wahr
2	wahr

Bild 1 Ergebnisse des Kurzprogramms zu den Zahlenwerten

```
10 CLS: PRINT TAB(12); "## ANDOR ##": PRINT
20 DIM A(3), A$(3)
30  FOR X=1 TO 2 STEP 0
40   INPUT"Zahlen ="; A(0), A(1)
50   A(2) = A(0) OR A(1): A(3) = A(0) AND A(1)
60    FOR I=0 TO 3: B=32768: A=A(I)
70     IF A<0 THEN A = 32768 + A: A$(I)="1": ELSE A$(I)="0"
80      FOR J=1 TO 15: B=B/2
90       IF A>=B THEN A=A-B: A$(I)=A$(I)+"1": ELSE A$(I)=A$(I)+"0"
100      IF J=3 OR J=7 OR J=11 THEN A$(I) = A$(I) + " "
110     NEXT
120   NEXT
130   FOR I=0 TO 3
140    PRINT A(I), A$(I)
150    IF I=1 THEN PRINT"    A OR B"
160    IF I=2 THEN PRINT"    A AND B"
170   NEXT: PRINT
180 NEXT
```

Bild 2 Programm ANDOR

```
              ## ANDOR ##

 137          56
 137         0000 0000 1000 1001
 56          0000 0000 0011 1000
    A OR B
 185         0000 0000 1011 1001
    A AND B
 8           0000 0000 0000 1000


-1234         3456
-1234        1111 1011 0010 1110
 3456        0000 1101 1000 0000
    A OR B
-82          1111 1111 1010 1110
    A AND B
 2304        0000 1001 0000 0000
```

Bild 3 Mit ANDOR ermittelte Werte

und auch so dargestellt. Das Vorzeichen wird als höchstwertiges Bit hinzugefügt. Mit diesem Programm kann man sich dann überzeugen, daß AND und OR genau bitweise diese Zahlen verwenden. Dies bedeutet, daß AND und OR Zahlen bitweise kombinieren und dabei die Zahlen wie vorzeichenbehaftete 16-Bit-Binärzahlen nutzen und daraus eine solche neue Zahl erzeugen. Dies gilt selbst für Gleitkommazahlen. Zahlen größer als 32768 bzw. kleiner als −32767 führen zu Fehlermeldungen.
Mit dieser Methode ist es nun leicht möglich, einzelne Bit-Operationen auszuführen:

B = A AND 8

weist B nur das Bit 4 von A zu.

B = A OR 256

übergibt A an B und setzt zusätzlich Bit 8. Diese Operationen sind insbesondere mit den Ein- und Ausgaben über Ports, also bezüglich INP und OUT recht interessant.
In anderen BASIC-Dialekten werden zuweilen nur 8 Bit ohne Vorzeichen oder eine andere Auswahl verwendet.

KC85/3-BASIC-Tip Maschinenprogramme

Im gleichen Heft auf Seite 317 wird ein Verfahren vorgestellt, Maschinenprogramme im BASIC-Programm unterzubringen. Im folgenden soll ein Verfahren gezeigt werden, welches gegenüber dem auf Seite 317 beschriebenen in bestimmten Fällen Vorteile bieten kann. Hierbei werden das Maschinenprogramm oder auch die Daten (z. B. Zeichengenerator) an das fertige BASIC-Programm angehängt und mit diesem zusammen abgespeichert und auch wieder geladen.
In den Zellen 3D7/3D8H steht die erste freie Adresse nach dem BASIC-Programm. Ab dieser Adresse beginnt der BASIC-Interpreter, seine Variablen abzulegen. Soll in diesem Bereich das Maschinenprogramm abgelegt werden, muß der Zeiger (3D7/3D8H) um die entsprechende Länge weiter nach höheren Adressen verlagert werden. Die Änderung dieses Zeigers sowie die Eingabe des Programmes erfolgen vom Betriebssystem mit dem MODIFY-Kommando.
Der Vorteil dieses Verfahrens besteht darin, daß im Maschinenprogramm 00-Bytes enthalten sein dürfen, was bei Zeichengeneratoren stets der Fall ist. Weiterhin erfolgt kein Auflisten dieses Bereiches.
Der Nachteil ist, daß Änderungen im BASIC-Programm eine Verschiebung des Maschinenprogrammes bedingen. Ein Aufruf der RENUMBER-Anweisung ist ebenfalls nicht mehr möglich. Zu beachten ist, daß bei voll ausgebautem Speicher dieser Bereich ggf. „hinter" dem Bildwiederholspeicher liegt.

K.-D. Kirves

Literatur
/1/ Kirves, K.-D.: KC85/3-BASIC-Tip Maschinenprogramme. Mikroprozessortechnik, Berlin 1 (1987) 10, S. 317

Digital-Ein-/Ausgabemodul für KC 85/2 und /3

Digital IN/OUT M001

Klaus-Dieter Kirves, Bernd Schenk,
Karsten Schiwon,
VEB Mikroelektronik
„Wilhelm Pieck" Mühlhausen

Die Kleincomputer KC 85/2 und KC 85/3 können durch den Einsatz von Erweiterungsmodulen in ihrem Leistungsumfang ausgebaut werden. Bis jetzt stehen dem Anwender die Module M003 – V24, M005 – USER, M006 – BASIC (für KC 85/2), M007 – ADAPTER, M012 – TEXOR, M022 – EXPANDER RAM, M025 – USER PROM 8K, M026 – FORTH, M027 – DEVELOPMENT und M011 64 KByte RAM zur Verfügung.
Mit dem Modul M001 – DIGITAL IN/OUT wurde eine weitere sinnvolle Systemerweiterung geschaffen, die es ermöglicht, die Kleincomputer auch zur Lösung von Aufgaben der Erfassung digitaler Daten, ihrer Bewertung und Ausgabe zur Steuerung peripherer Geräte zu verwenden.
Auch für die Lehre und Weiterbildung bestehen mit dem Modul M001 bessere Möglichkeiten, den Anforderungen gerecht zu werden. So können Experimente zu Steuerungen mit Hilfe des Computers demonstriert werden. Im Rahmen des Unterrichtsfaches „Automatisierungstechnik" ist der Einsatz des Kleincomputers mit dem Modul M001 zur Simulation von Prozeßsteuerungen in der Berufsausbildung vorgesehen.
Eine weitere Einsatzmöglichkeit des Moduls besteht in der Ansteuerung von peripheren Geräten mit CENTRONICS – Schnittstellen. Dabei ist es im System der Kleincomputer KC 85/2 und KC 85/3 möglich, mehrere Module gleichen Types quasi gleichzeitig zu betreiben, indem der jeweils gewünschte Modul zur Ein-/Ausgabe selektiert werden kann.

Aufbau des Moduls M001 DIGITAL IN/OUT

Zur Realisierung der Ein-/Ausgabe von digitalen Informationen dienen die Schaltkreise UB 855 (PIO) und UB 857 (CTC). Die Belegung des Steckverbinders in Tafel 1 zeigt, daß beide Ports des UB 855 und 2 Kanäle des UB 857 für den Datenaustausch mit anderen Geräten zur Verfügung stehen. Die zwei verbleibenden Kanäle des UB 857 sind auf dem Modul kaskadiert und können als Softwareuhr bzw. unabhängig als Zeitgeber genutzt werden.
Der Modul beinhaltet eine Modulsteuerung, die den Modul in das System des Computers einbindet. Mit der Modulsteuerung wird der Modul im System online oder offline geschaltet. Ein-/Ausgaben zur Programmierung der Systemschaltkreise oder Datenein-/-ausgaben sind nur im eingeschalteten Zustand (online) des Moduls möglich.
Der Adreßdecoder selektiert die Adressen der Systemschaltkreise UB 855 und UB 857.

Tafel 1 Belegung des Steckverbinders

Kontakt	Bezeichnung	Anmerkungen
1A, 1B	Masse	
1A	PIO A0	Datenein-/
2B	PIO B0	ausgänge
3A	PIO A1	"
3B	PIO B1	"
4A	PIO A2	"
4B	PIO B2	"
5A	PIO A3	"
5B	PIO B3	"
6A	PIO A4	"
6B	PIO B4	"
7A	PIO A5	"
7B	PIO B5	"
8A	PIO A6	"
8B	PIO B6	"
9A	PIO A7	"
9B	PIO B7	"
10A	PIO /ASTB	Handshake-
10B	PIO /BSTB	signale
11A	PIO ARDY	"
11B	PIO BRDY	"
12A	CTC C/TRGO	Eingang CTC
12B	CTC ZC/TO0	Ausgang CTC
13A	CTC C/TRG1	Eingang CTC
13B	CTC ZC/TO1	Ausgang CTC

Tafel 2 Adressen der Systemschaltkreise

Bezeichnung	*I/O-Adresse*
CTC Kanal 0	*0*
CTC Kanal 1	*1*
CTC Kanal 2	*2*
CTC Kanal 3	*3*
PIO A Daten	*4*
PIO B Daten	*5*
PIO A Steuerregister	*6*
PIO B Steuerregister	*7*

Da diese Adressen für alle Module M001 gleich sind, werden Ein-/Ausgaben auf gleichartige Module bzw. Systemschaltkreise dieser Module durch die Signale MEI und MEO des KC-85-Systems verhindert. Diese Signale bilden eine Prioritätskette. Das MEO-Signal des Moduls, der im Computer die niedrigste Steckplatznummer hat, hat die höchste Priorität. Ist dieser Modul eingeschaltet, werden durch das MEO-Signal Ausgaben auf gleiche Module mit höheren Steckplatzadressen verhindert. Ist dieser Modul abgeschaltet, ist MEI gleich MEO, und es kann der nächste gleichartige Modul angesprochen werden. Eine Adreßtabelle der Schaltkreise des M001 ist in Tafel 2 angegeben.
Die Interruptlogik bindet die Schaltkreise UB 855 und UB 857 in das Interruptsystem des Computers ein.
Für Interrupts ist der UB 857 (CTC) höher priorisiert.
Das Blockschaltbild des Moduls ist in Bild 1 dargestellt.

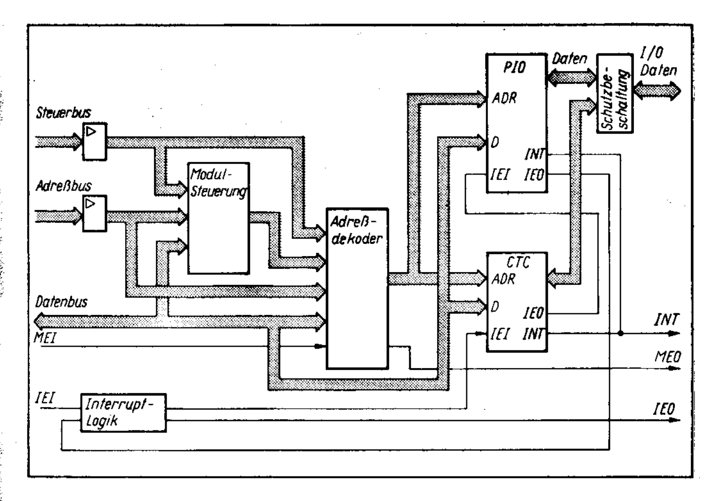


Bild 1 Blockschaltbild des Moduls M001 DIGITAL IN/OUT

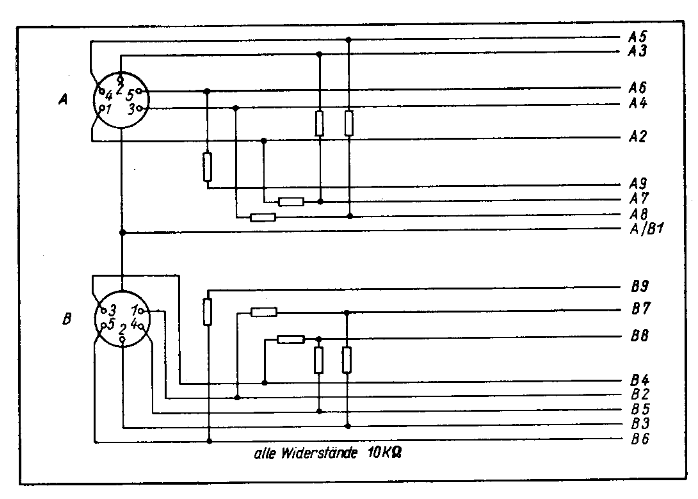


Bild 3 Schaltung des Steuerknüppeladapters

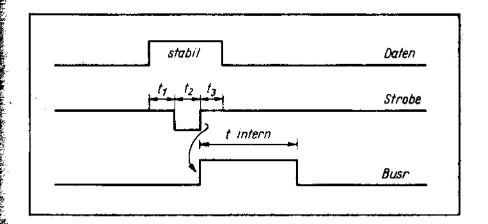


Bild 2 Zeitablauf der Datenübertragung, dabei ist t_{intern} die druckerinterne Verarbeitungszeit und es gilt: $t_1 \geqq 1\mu s$, $t_2 \geqq 1\mu s$ und $t_3 \geqq 1\mu s$

Dipl.-Ing. Klaus-Dieter Kirves studierte von 1976 bis 1981 an der Technischen Hochschule Karl-Marx-Stadt in der Fachrichtung Technische Kybernetik und Automatisierung. Seitdem ist er als Entwicklungsingenieur im VEB Mikroelektronik Wilhelm Pieck Mühlhausen tätig, zuerst in der Taschenrechnerentwicklung und jetzt in der Systemprogrammierung bei der Entwicklung der Kleincomputer.

Dipl.-Ing. Bernd Schenk studierte von 1972 bis 1976 an der Technischen Hochschule Ilmenau in der Fachrichtung Informationstechnik und theoretische Elektrotechnik. Er ist Entwicklungsingenieur im VEB Mikroelektronik Wilhelm Pieck Mühlhausen, arbeitete zuerst in der Taschenrechnerentwicklung und ist jetzt im Hardware-Entwurf bei der Entwicklung der Kleincomputer tätig.

Dipl.-Ing. Karsten Schiwon studierte von 1978 bis 1984 am Leningrader Elektrotechnischen Institut in der Fachrichtung Elektronische Rechengeräte und -anlagen. Seit Abschluß des Studiums ist er als Entwicklungsingenieur im VEB Mikroelektronik Wilhelm Pieck Mühlhausen in der Hard- und Softwareentwicklung der Kleincomputer tätig.

Die technischen Daten lauten:

Anzeige online/offline	
Modulsteuerung	
Modulkennung	EF
parallele Ein-/Ausgabekanäle	2
je Kanal Datenleitungen	8
je Kanal Quittungsleitungen	2
Zähler – Ein-/Ausgänge	2
Zählerkanäle	3
oder Zeitgeberkanäle	4

Prioritäts- und Interruptlogik
Schutzbeschaltung aller Ein- und Ausgänge im Spannungsbereich −3V bis +8V
Eingangspegel:
low 0–0,4V, high 2,4–5V
Ausgangspegel TTL für maximal eine TTL-Last

Beispiele

Als erstes Einsatzbeispiel des Moduls M001 DIGITAL IN/OUT soll eine Druckerschnittstelle, die mit dem Entwicklungssystem-Modul M027 erstellt wurde, dienen.
Diese Schnittstelle ermöglicht die bitparallele byteserielle Ausgabe von ASCII-Zeichen an einen Drucker mit CENTRONICS-Schnittstelle und wurde unter anderem mit einem Drucker K6312 getestet.
Für das Interface gelten folgende Übertragungsbedingungen:

– der Computer übergibt ein ASCII-Zeichen parallel auf den Datenleitungen
– sind die Daten stabil, wird auf der Leitung STROBE ein Low-Impuls ausgegeben
– der Drucker schaltet die Leitung BUSY auf High-Pegel und zeigt so den Datenempfang an
– der Drucker schaltet nach Datenübernahme die BUSY-Leitung auf low und zeigt die Empfangsbereitschaft an.

In Bild 2 ist ein Zeitablauf der Datenübertragung dargestellt.
Die Schnittstellenleitungen für das CENTRONICS-Interface sind in Tafel 3 aufgeführt. Diese Leitungen sollten eine Länge von 1 m nicht überschreiten. Weiterhin ist es zu empfehlen, die Datenleitungen jeweils mit einer Masseleitung zu verdrillen. Es sind sowohl die Kontaktbelegungen für den 39poligen Steckverbinder, wie er beim K6312 zum Einsatz kommt, als auch für den international verbreiteten 36poligen Amphenol-Steckverbinder angegeben.
In Tafel 4 ist das zugehörige Assemblerlisting mit Kommentaren dargestellt. Das Programm gestattet sowohl den zeichenweisen Ausdruck über den Anwenderkanal 2, dies ist im BASIC über PRINT#2 bzw., LIST#2, als auch ein Mitprotokollieren aller Bildschirmausgaben auf den Drucker. Das Mitprotokollieren wird über SHIFT CLEAR-Taste ein- bzw. ausgeschaltet. Bei der Initialisierung des Programmes kann der Modulschacht angegeben werden, in welchem der Modul M001 steckt. Weiterhin kann ausgewählt werden, über welchen Anwenderkanal ausgegeben werden soll. Dies ist z.B. sinnvoll, wenn gleichzeitig im System auch eine V.24-Schnittstelle eingesetzt wird. Der Aufruf des Programmes erfolgt aus der Menüeingabe des Betriebssystems.
Folgendes Beispiel verdeutlicht dies:
%CENTRON 8 2
Hier wird der Modul im Schacht mit der Adresse 8 für den Anwenderkanal 2 (im BASIC#2) initialisiert. Dies entspricht der Initialisierung in Tafel 4.
Steht ein DEVELOPMENT-Modul M027 zur Verfügung, so kann das Listing eingegeben und übersetzt werden. Anderenfalls kann auch eine Eingabe des Maschinenkodes mittels des CAOS-Kommandos MODIFY ab Adresse BA00 erfolgen. In beiden Fällen wird das Maschinenprogramm als selbststartendes Programm auf Kassette abgespeichert. Die Kommandoeingabe dafür lautet:
%SAVE BA00 BB00 BA02
NAME: CENTRONICOM

Tafel 3 Verbindungen für CENTRONICS-Schnittstelle

Kontakt/Rückleitung (39polig)		Kontakt/Rückleitung (AMPHENOL)		Bezeichnung	Kontakt (M001)
B2	C2	1	19	/STROBE	B2
B5	C5	2	20	DAT1	A1
B6	C6	3	21 •	DAT2	A2
B7	C7	4	22	DAT3	A3
B8	C8	5	23	DAT4	A4
B9	C9	6	24	DAT5	A5
B10	A10	7	25	DAT6	A6
B11	A11	8	26	DAT7	A7
B12	A12	9	27	DAT8	A8
C11	C13	11	29	BUS	B4
A13		(beliebige Rückleitung)		0V	A/B1

```
3000          ;*************************************
3000                 ORG     0BA00H
BA00          ; CENTRONIC- TREIBER
BA00          ;         4.9.86
BA00          ; K.SCHIWON & K.-D. KIRVES
BA00          ;     VEB MIKROELEKTRONIK
BA00          ;         MUEHLHAUSEN
BA00          ;*************************************
BA00          ;VEREINBARUNGEN:
BA00          CAOS    EQU     0F003H
BA00          MODU    EQU     26H     ;UP-NR. MODUL
BA00          CRT     EQU     0       ;UP-NR. CRT
BA00          ;SYSTEMZELLEN CAOS
BA00          ARGN    EQU     0B781H
BA00          STBT    EQU     0B7A2H  ;STEUERBYTE
BA00          OUTAB   EQU     0B7B9H  ;OUT-CHANAL
BA00          UOUT1   EQU     0B7BEH  ;USER OUT 1
BA00          UOUT2   EQU     0B7C4H  ;  "    2
BA00          HCADR   EQU     0B799H  ;SHIFT CLEAR
BA00          CRR     EQU     0DH
BA00          ;
BA00          MKENN   EQU     0EFH    ;MODULKENNBYTE
BA00          PIOA    EQU     04H     ;PIO KANAL A DAT
BA00          PIOAC   EQU     06H     ;    STEUERWORT
BA00          PIOB    EQU     05H     ;PIO B DATEN
BA00          PIOBC   EQU     07H     ;    STEUERWORT
BA00          ;
BA00          ;*************************************
BA00          ;PARAMETERUEBERGABE BEI AUFRUF:
BA00          ;CENTRON MODULSCHACHT USER-OUT
BA00          ;        8 u.s.w.       2/3
BA00          ;*************************************
BA00 00       HCPZ:   DEFB    0       ;USER-KANAL
BA01          ;
BA01          ; MODULSCHACHT 8
BA01 08       MODSCH: DEFB    8       ;MODULSCHACHT
BA02          ;-------------------------------------
BA02 C324BA           JP      START   ;SELBSTSTARTADR.
BA05          ;-------------------------------------
BA05 7F7F             DEFW    7F7FH   ;PROLOG
BA07 43454E54         DEFM    'CENTRON'
BA0E 01               DEFB    1
BA0F 3A81B7           LD      A,(ARGN)
BA12 FE02             CP      2
BA14 380E             JR      C,START
BA16 7D               LD      A,L     ;ARG1,LOW
BA17 3201BA           LD      (MODSCH),A      ;SCHACHT
BA1A 7B               LD      A,E     ;USEROUT
BA1B D602             SUB     2
BA1D 3811             JR      C,ERROR
BA1F E601             AND     1
BA21 3200BA           LD      (HCPZ),A
BA24          ; SELBSTSTARTADRESSE
BA24          ;+++++++++++++++++++++++++++++++++++++
BA24 3A01BA   START:  LD      A,(MODSCH)
BA27 47               LD      B,A
BA28 0E80             LD      C,80H
BA2A ED78             IN      A,(C)
BA2C FEEF             CP      MKENN
BA2E 2805             JR      Z,ST1
BA30 CD03F0   ERROR:  CALL    CAOS    ;FEHLERABBRUCH
BA33 19               DEFB    19H
BA34 C9               RET
BA35          ;
BA35 68       ST1:    LD      L,B
BA36 3E02             LD      A,2
BA38 1601             LD      D,1
BA3A 5A               LD      E,D
BA3B CD03F0           CALL    CAOS    ;MODULSWITCH
BA3E 26               DEFB    MODU
BA3F 3EFF             LD      A,0FFH
BA41 D306             OUT     PIOAC,A ;BIT E/A
BA43 3E00             LD      A,0
BA45 D306             OUT     PIOAC,A ;ALLE BIT OUT
BA47 3EFF             LD      A,0FFH
BA49 D307             OUT     PIOBC,A ;BIT E/A
BA4B 3E04             LD      A,4
BA4D D307             OUT     PIOBC,A ;BIT0 OUT
BA4F 3E01             LD      A,1
BA51 D305             OUT     PIOB,A  ;STROBE PASSIV
BA53 215FBA           LD      HL,HCOP
BA56 2299B7           LD      (HCADR),HL
BA59 E5               PUSH    HL
BA5A D5               PUSH    DE
BA5B 1830             JR      HAC01
BA5D          ;
BA5D 02       ZWEI:   DEFB    2
BA5E 03       DREI    DEFB    3
BA5F          ;
BA5F          ;PROTOKOLLFUNKTION UEBER SHIFT CLEAR
BA5F          ;ODER BILDSCHIRMAUSDRUCK
BA5F E5       HCOP:   PUSH    HL
BA60 D5               PUSH    DE
BA61 2AB9B7           LD      HL,(OUTAB)      ;KANAL
BA64 34               INC     (HL)
BA65 35               DEC     (HL)
BA66 2025             JR      NZ,HAC01        ;CRT ->
BA68 1195BA           LD      DE,PROT         ;CRT+KAN
BA6B 215DBA           LD      HL,ZWEI
BA6E 3A00BA           LD      A,(HCPZ)
BA71 A7               AND     A
BA72 2803             JR      Z,HAC03         ;USER2->
BA74 215EBA           LD      HL,DREI
BA77 3A00BA   HAC03:  LD      A,(HCPZ)
BA7A A7               AND     A
BA7B 2806             JR      Z,HAC02
BA7D ED53C4B7         LD      (UOUT2),DE      ;EINTR.3
BA81 1804             JR      HAC04
BA83          ;
BA83 ED53BEB7 HAC02:  LD      (UOUT1),DE      ;EINTR.2
BA87 22B9B7   HAC04:  LD      (OUTAB),HL
BA8A D1       HAPOP:  POP     DE
BA8B E1               POP     HL
BA8C C9               RET
BA8D          ;
BA8D 2199BA   HAC01:  LD      HL,ZCRT         ; CRT
BA90 119BBA           LD      DE,SDD
BA93 18E2             JR      HAC03
BA95          ;PROTOKOLLAUSGABE
BA95 F5       PROT:   PUSH    AF
BA96 CD03F0           CALL    CAOS    ;BILDSCHIRMAUSG.
BA99 00       ZCRT:   DEFB    00H
BA9A F1               POP     AF
BA9B          ;*************************************
BA9B          ; ZEICHENAUSGABE ZEICHEN IN A
BA9B FE09     SDD:    CP      09H     ;RIGHT?
BA9D 280D             JR      Z,SD4
BA9F F5               PUSH    AF
BAA0 3AA2B7           LD      A,(STBT)
BAA3 CB5F             BIT     3,A     ;INTERPRET?
BAA5 280F             JR      Z,SD9   ;YES
BAA7 F1               POP     AF
BAA8 FE7F             CP      7FH     ;AUSBLENDEN 7FH
BAAA 2002             JR      NZ,SD3
BAAC 3E20     SD4:    LD      A,20H
BAAE FE20     SD3:    CP      20H     ;SPACE
BAB0 301A             JR      NC,SD1
BAB2 3E5F             LD      A,5FH
BAB4 1816             JR      SD1
BAB6 F1       SD9:    POP     AF
BAB7 FE0D             CP      0DH
BAB9 2011             JR      NZ,SD1
BABB          ;ZEILENPOSIT 8* SPACE
BABB C5       SD0:    PUSH    BC
BABC F5               PUSH    AF
BABD CDCCBA           CALL    SD1
BAC0 0608             LD      B,8
BAC2 3E20     SDLOP:  LD      A,20H
BAC4 CDCCBA           CALL    SD1
BAC7 10F9             DJNZ    SDLOP
BAC9 F1               POP     AF
BACA C1               POP     BC
BACB C9               RET
BACC          ;DIREKTE BYTEAUSGABE
BACC C5       SD1:    PUSH    BC      ;AUSGABE DATEN
BACD F5               PUSH    AF
BACE DB05     SDA:    IN      A,PIOB
BAD0 CB57             BIT     2,A
BAD2 2808             JR      Z,SDB   ;BUSY?
BAD4 3E05             LD      A,5
BAD6 CD03F0           CALL    CAOS
BAD9 14               DEFB    14H     ;WARTEN
BADA 18F2             JR      SDA
BADC          ;
BADC F1       SDB:    POP     AF
BADD D304             OUT     PIOA,A  ;DATEN
BADF 3E00             LD      A,0
BAE1 D305             OUT     PIOB,A  ;STROBE AKTIV
BAE3 3E01             LD      A,1
BAE5 D305             OUT     PIOB,A  ;STROBE PASSIV
BAE7 C1               POP     BC
BAE8 C9               RET
BAE9          ;
BAE9          ;          E N D E

ERRORS: 0000
```

Tafel 4 Assemblerlisting der CENTRONICS-Schnittstelle (mit M027 Development erstellt)

Alle weiteren Informationen zum Programm können dem Listing entnommen werden.

Das zweite Beispiel soll der Anschluß von zwei zum Kleincomputer KC 85/1 des VEB Robotron Meßelektronik „Otto Schön" Dresden gehörenden Steuerknüppeln an den KC 85/3 sein.

In Bild 3 ist ein Adapter zum Anschluß der mit Diodensteckern versehenen Steuerknüppel an den Modul M001 dargestellt. Die Steuerknüppel besitzen jeweils vier Schaltkontakte für die vier Grundrichtungen und eine zusätzliche Auslösetaste. Da bei Betätigung der Steuerknüppel in die jeweiligen Zwischenstellungen zwei Kontakte geschlossen werden, können 9 Stellungen des Steuerknüppels ausgewertet werden. Bild 4 gibt die Zusammenhänge zwischen Stellung des Steuerknüppels und Funktionswert, wie er vom Hersteller des KC 85/1 angegeben wird, wieder. Zur Realisierung dieses Anschlusses an den M001 wird je Steuerknüppel ein PIO-Kanal im BIT-Ein-Ausgabemodus (Mode 3) verwendet. Da aus dem Modul die Versorgungsspannung nicht mit herausgeführt ist, müssen einige PIO-Ausgänge den High-Pegel liefern. Um die zugehörigen Ausgänge nicht zu sehr zu belasten, wurde die Schaltung so gewählt, daß je Ausgang immer nur ein Widerstand gegen Masse geschaltet werden kann.

In den Anwenderprogrammen für die Steuerknüppel muß zuerst die Programmierung der beiden PIO-Kanäle erfolgen:

① Assemblerprogramm

```
LD  A,0FFH      ;Mode3
OUT (06H),A     ;Steuerwort PIO A
OUT (07H),A     ;Steuerwort PIO B
LD  A,1FH       ;E/A Maske
OUT (06H),A
OUT (07H),A
LD  A,0E0H      ;Aktivierung der Widerstände
OUT (04H);A     ;Datenwort PIO A
OUT (05H);A     ;Datenwort PIO B
```

② BASIC-Programm

```
10 OUT6,255:OUT7,255:OUT6,31:OUT7,31
20 OUT4,240:OUT5,240
```

Bei der Eingabe der Funktionswerte ist zu beachten, daß diese negiert vorliegen, die Bits 5–7 gesetzt sind und ausgeblendet werden müssen.

Das Lesen des Steuerknüppels A erfolgt:

① Assemblerprogramm

```
IN   A,(04H)    ;Datenwort PIO A
CPL             ;Negation
AND 1FH         ;Ausblenden Bits 5–7
```

② BASIC-Programm

```
100 KA=NOT(INP(4)AND31):REM KA Funktionswert
```

Der Aufbau des Adapters kann auf einer Universalleiterplatte erfolgen, welche zwei fünfpolige Diodenbuchsen sowie die 10 Widerstände enthält. Die Verbindung zum Modul wird über Flachbandkabel zum mitgelieferten direkten Steckverbinder realisiert.

Der Eigenbau solch einfacher Steuerknüppel ist z. B. mit Mikrotastern relativ einfach möglich.

Die Steuerknüppel können, z. B. bei der Eingabe in Grafikprogrammen, die Programmbedienung wesentlich erleichtern.

Stellung	Funktionswert dezimal	dual
	1	00001
	2	00010
	4	00100
	8	01000
	5	00101
	6	00110
	9	01001
	10	01010
Taste	>=16	1XXXX

Bild 4 Funktionswerte der Steuerknüppel

Standard-Interfaces über den User-Port des KC 85/1

Dr. Frank Schwarzenberg
Zentralinstitut für Kernforschung
der AdW der DDR
Rossendorf, Dresden

Für Geräte der Leistungsklasse des KC 85/1, die bereits in der DDR eine große Verbreitung erfahren haben, werden i. a. keine peripheren Geräte mitgeliefert. Häufig besteht deshalb der Wunsch, bereits vorhandene Geräte wie Drucker, Lochbandleser und -stanzer, Teletypes, Fernschreiber o. ä. an den Kleincomputer anzuschließen. Von Robotron wird bisher nur der Anschluß spezieller Drucker (K6311, K6303) unterstützt. In dem Beitrag werden Möglichkeiten dargestellt, wie ohne spezielle Zusatzmodule und mit geringstem Hardwareaufwand in vielen Fällen ein Anschluß der o. g. Geräte durch Nutzen des User-Ports des KC85/1 realisierbar ist.
Die Problematik des Einbindens eigener Gerätetreiber in das Betriebssystem des KC 85/1 wird ausführlich erläutert.
Die in dem Beitrag angeführten Programmbeispiele sind erprobt und ermöglichen u. a. das Betreiben solcher Drucker wie EPSON LX86 (V24/DTR-Protokoll) oder K6303 (TD40) am KC85/1 **ohne** *Druckermodul.*

1. User-Port des KC85/1

Bild 1 zeigt den für den Anwender frei verfügbaren User-Port des KC 85/1. Er ist über eine 15polige EFS-Buchse an der rechten Seite des Gerätes dem Nutzer zugänglich und besteht aus einem PIO-Port und einem CTC-Kanal. Diese Peripherieschaltkreise sind über folgende E/A-Adressen ansprechbar:
PIO-Daten: 89H
PIO-Control: 8BH
CTC1: 81H

Bei der Nutzung dieser Schnittstelle müssen unbedingt die zulässigen Pegel und Lastbedingungen von PIO und CTC beachtet werden, da diese Anschlüsse im KC 85/1 **nicht** gegen Überlastung bzw. Zerstörung geschützt sind.

2. Software-Realisierung von Standard-Interfaces

Aus Bild 1 geht hervor, daß dem Anwender am User-Port 8 freiprogrammierbare Leitungen und jeweils zwei Ein- und Ausgabeleitungen mit speziellem Verhalten zur Verfügung stehen. Diese Belegung erlaubt bei konsequenter Ausnutzung die Realisierung so gängiger Standard-Interfaces wie SIF-1000, V24(RS 232C) und CENTRONICS derart, daß viele Geräte mit diesen Schnittstellen **direkt** an den User-Port angeschlossen werden können.
Die notwendigen Hardware-Arbeiten beschränken sich – wie die folgenden Beispiele zeigen – auf das Anfertigen eines Kabels und evtl. erforderliche Pegelanpassungen (V24).
Es sei darauf hingewiesen, daß die Beschreibung der Standard-Interfaces nur soweit erfolgt, wie das für den konkreten Geräteanschluß notwendig ist, und somit keinen allgemeingültigen Charakter besitzt.

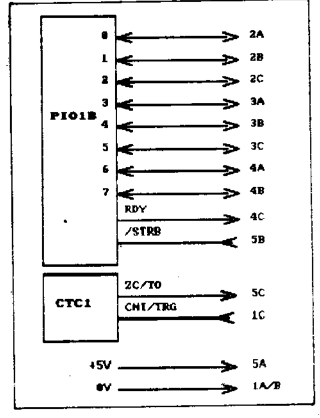


Bild 1 User-Port des KC–85/1

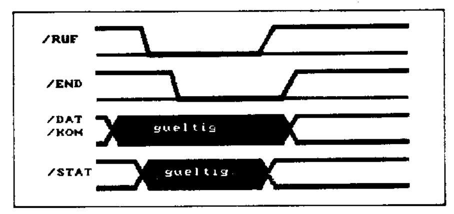


Bild 2 Signalverhalten beim SIF-1000-Interface

2.1. *SIF-1000*

SIF-1000 ist ein in der DDR noch weit verbreitetes Standard-Interface. Lochbandleser und -stanzer (z. B. daro 1215) sowie ältere Druckertypen (z. B. SD 1156) sind mit diesem Interface ausgestattet. Bevor man ein derartiges Gerät anschließt, sind folgende gerätespezifischen Parameter zu prüfen:

– Pegelverhältnisse (TTL, KME3 (12V), KME 20 etc.) und Lastbedingungen

Geräte mit TTL- oder KME3-Pegel ermöglichen häufig den direkten Anschluß an den User-Port. Bei KME3 genügt meist das „Klemmen" der vom Gerät kommenden Signale über Dioden gegen +5 V. Die vom User-Port abgehenden TTL-Signale werden meist von den Geräten sicher erkannt. Dies sollte jedoch anhand der Gerätebeschreibungen im konkreten Fall überprüft werden.

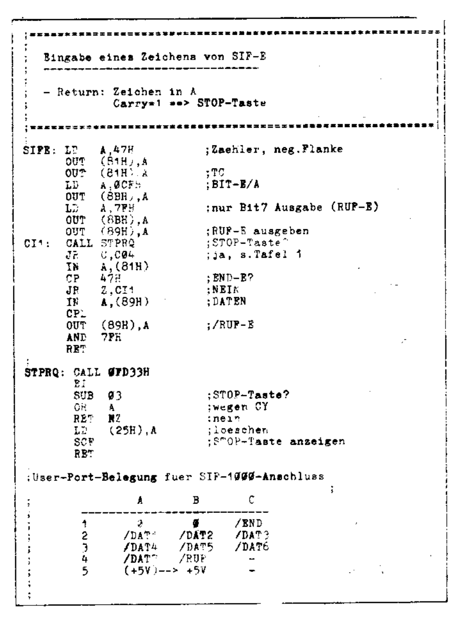
– Paritätsprüfung

Bei SIF-1000-Geräten wird das 8. Datenbit oft als Paritätsbit verwendet. Das hier angegebene Beispiel einer reinen Software-Realisierung des SIF-1000-Signalspiels nutzt das 8. PIO-Bit für die Erzeugung des RUF-Signales. Ein Paritätsbit kann somit nicht gebildet

Tafel 1 Treiberroutine für SIF-1000-Ausgabe

```
;==============================================================
;
;  SIF-1ØØØ Ein-/Ausgabe ueber User-Port
;  -------------------------------------
;  Voraussetzungen:
;- RUF wird ueber PIO-Bit 7 gebildet ==> nur 7 Datenbits
;  und keine Paritaet moeglich!
;- Die Flanken von END-A werden mit dem CTC-KANAL erfasst
;- keine KOM-Ausgabe
;- keine STAT-Auswertung
;
;==============================================================
;
;  Ausgabe eines Zeichens an SIF-A
;  -------------------------------
;- Zeichen im C-Register
;- Return: Carry=1 ==> Stop-Taste
;
;==============================================================
;
SIFA: LD   A,47H            ;DI,Zaehler,neg.Flanke,TC,RESET
      OUT  (81H),A          ;INIT CTC-1
      OUT  (81H),A          ;TC=47H (willkuerlich)
      LD   A,ØFH            ;PIO1/B:BYTEAUSGABE
      OUT  (8BH),A
      LD   A,C
      CPL                   ;Bei SIF-1ØØØ Daten low active
      SET  7,A              ;ohne RUF-A
      OUT  (89H),A          ;Zeichen ausgeben
      NOP
      NOP                   ;Einschwingen lassen
      RES  7,A              ;RUF-A
      OUT  (89H),A          ;Zeichen gueltig
CØ1:  CALL STPRQ            ;STOP-Taste?
      JR   NC,CØ3           ;nein
CØ4:  CPL                   ;ja
      OUT  (89H),A          ;RUF-A zuruecknehmen
      RET
CØ3:  IN   A,(81H)
      CP   47H              ;END-A?
      JR   Z,CØ1            ;nein, warten
      LD   A,57H            ;s.o.,aber pos. Flanke
      OUT  (81H),A
      OUT  (81H),A          ;TC=57H
      LD   A,ØFFH           ;/RUF-A
      OUT  (89H),A
CØ2:  CALL STPRQ            ;STOP-Taste?
      JR   C,CØ4            ;ja
      IN   A,(81H)
      CP   57H              ;/END-A?
      JR   Z,CØ2            ;nein, warten
      OR   A
      RET
;
```

Tafel 2 Treiberroutine für SIF-1000-Eingabe



```
;==============================================================
;
;  Eingabe eines Zeichens von SIF-E
;  --------------------------------
;
;  - Return: Zeichen in A
;            Carry=1 ==> STOP-Taste
;
;==============================================================
;
SIFE: LD   A,47H            ;Zaehler, neg.Flanke
      OUT  (81H),A
      OUT  (81H),A          ;TC
      LD   A,ØCFH           ;BIT-E/A
      OUT  (8BH),A
      LD   A,7FH            ;nur Bit7 Ausgabe (RUF-E)
      OUT  (8BH),A
      OUT  (89H),A          ;RUF-E ausgeben
CI1:  CALL STPRQ            ;STOP-Taste?
      JR   C,CØ4            ;ja, s.Tafel 1
      IN   A,(81H)
      CP   47H              ;END-E?
      JR   Z,CI1            ;NEIN
      IN   A,(89H)          ;DATEN
      CPL
      OUT  (89H),A          ;/RUF-E
      AND  7FH
      RET
;
STPRQ: CALL ØFD33H
       EI
       SUB  Ø3              ;STOP-Taste?
       OR   A               ;wegen CY
       RET  NZ              ;nein
       LD   (25H),A         ;loeschen
       SCF                  ;STOP-Taste anzeigen
       RET

;User-Port-Belegung fuer SIF-1ØØØ-Anschluss
;
;               A        B        C
;         ----------------------------
;         1     Ø        Ø       /END
;         2    /DAT1    /DAT2    /DAT3
;         3    /DAT4    /DAT5    /DAT6
;         4    /DAT7    /RUF      -
;         5    (+5V)--> +5V       -
;
```

Tafel 3 Treiberroutine für CENTRONICS-Drucker

```
;==============================================================
;
; CENTRONICS-Druckerschnittstelle ueber User-Port
;
;- 7 Datenbits verfuegbar
;- PIO-Bit7 wird zur Bildung des CENTRONICS-/STROBE-
;  Signals verwendet.
;- /ACKNLG vom Drucker wird ueber den CTC-Kanal erfasst.
;
;- Zeichen in Reg. C
;- Return: Carry=1 bei Stop-Taste
;
;==============================================================
;
CENTR: LD   A,ØCFH
       OUT  (8BH),A         ;Bit-Mode
       XOR  A               ;alles Ausgabe
       OUT  (8BH),A
       LD   A,57H           ;Zaehler,pos. Flanke
       OUT  (81H),A
       OUT  (81H),A         ;INIT CTC-1
       LD   A,C
       LD   B,8ØH
       OR   B               ;/STROBE=HIGH
       OUT  (89H),A         ;Zeichen ohne Strobe
       XOR  B               ;/STROBE=LOW
       OUT  (89H),A         ;Daten gueltig
       OR   B               ;/STROBE=HIGH
       OUT  (89H),A
CNTR1: CALL STPRQ           ;STOP-Taste?
       RET  C
       IN   A,(81H)
       CP   57H             ;/ACKNLG?
       JR   Z,CNTR1         ;nein, warten
       XOR  A
       RET
;
;
;User-Port-Belegung fuer CENTRONICS-Anschluss
;
;               A        B        C
;         ----------------------------
;         1     Ø        Ø       /ACK
;         2    DAT1     DAT2     DAT3
;         3    DAT4     DAT5     DAT6
;         4    DAT7    /STROBE    -
;         5    (+5V)==> +5V       -
;
```

werden. Die meisten Geräte gestatten aber das Abschalten der Paritätsprüfung.

– Nutzung der KOMmando- bzw. STATus-Signale

Die begrenzte Anzahl der Datenleitungen am User-Port gestattet nicht die Bildung der KOM- bzw. Auswertung der STAT-Signale des SIF-1000 ohne Zusatzhardware. In den meisten der hier interessierenden Anwendungsfälle kann auf die dynamische Nutzung dieser Leitungen verzichtet werden. Für die Datenübertragung müssen die Kommando-Signale normalerweise die folgende Belegung haben (sofern sie vom Gerät überhaupt ausgewertet werden):

KOM-1: 1
KOM-2: Ø
KOM-3: Ø

Diese Belegung kann – falls erforderlich – fest verdrahtet werden.
Über die max. 3 Statusleitungen werden u. U. bestimmte Gerätebedingungen wie Lochband- oder Papierende bzw. Gerätefehler gemeldet. Auf die Auswertung dieser Bedingungen kann unter Berücksichtigung der hier interessierenden Anwendung verzichtet werden.
Bild 2 zeigt das für SIF-1000 typische Signalverhalten. Über die Begleitsignale **RUF** und **END** wird ein sog. vollständiges Hand-shake realisiert.
Bei Beachtung der genannten Einschränkungen kann der Anschluß eines SIF-1000-Gerätes an den User-Port mit folgender Signal-Zuordnung erfolgen:

User-Port	SIF-1000-Gerät
PIO-0	/DAT-1
PIO-1	/DAT-2
PIO-2	/DAT-3
PIO-3	/DAT-4
PIO-4	/DAT-5
PIO-5	/DAT-6
PIO-6	/DAT-7
PIO-7	/RUF
CTC-CNT/TRG	/END

Die PIO-Signale RDY und /STRB werden hier nicht verwendet, da sie das für RUF und END geforderte Signalverhalten ohne Zwischenschalten von Zusatzelektronik nicht realisieren.
Für die Auswertung des END-Signals wurde der Zähler-Eingang des CTC-Kanales benutzt. Damit sind die in Tafel 1 und 2 angeführten Routinen zur softwaremäßigen Realisierung des Geräteanschlusses ohne Interruptbetrieb ausführbar.
Die angegebenen Treiber bilden das in Bild 2 dargestellte Signalverhalten für die Signale /DAT-1 bis /DAT-7, /RUF und /END nach. Um ein „Klemmen" des Systems bei nicht angeschlossenem Gerät oder anderen Fehlern zu verhindern, wurde eine Abfrage der STOP-Taste mit integriert, wodurch ein Zwangsaustritt aus den Routinen jederzeit möglich ist.
Die SIF-E-Routine wurde insofern verkürzt, als auf die Rückflanke des END-E-Signals nicht mehr gewartet wird. Das hat sich als vorteilhaft bei der Nutzung des Handlesers ZfK 4944-170/03 herausgestellt.

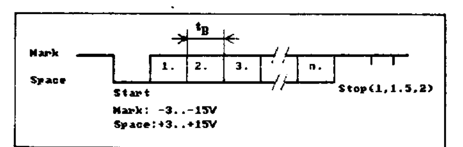


Bild 3 V24 – asynchrones Übertragungsprotokoll

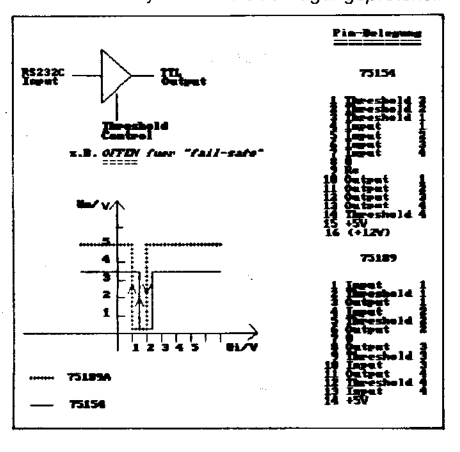


Bild 4 V24 – Treiber- und Empfängerschaltkreise im „failsafe"-Mode

Dr.-Ing. Frank Schwarzenberg studierte von 1970 bis 1974 an der TU Dresden in der Fachrichtung Informationstechnik. Von 1970 bis 1974 absolvierte er am VIK Dubna einen Arbeitsaufenthalt. 1980 erfolgte die Promotion zum Thema Rechnerkopplung zur Erfassung kernphysikalischer Experimentdaten. Seit 1982 ist unser Autor am ZfK Rossendorf tätig. Sein Arbeitsgebiet ist die Hard- und Softwareentwicklung für Klein- und Mikrorechentechnik (8-/16-Bit-Systeme).

2.2. V24 (RS 232 C)

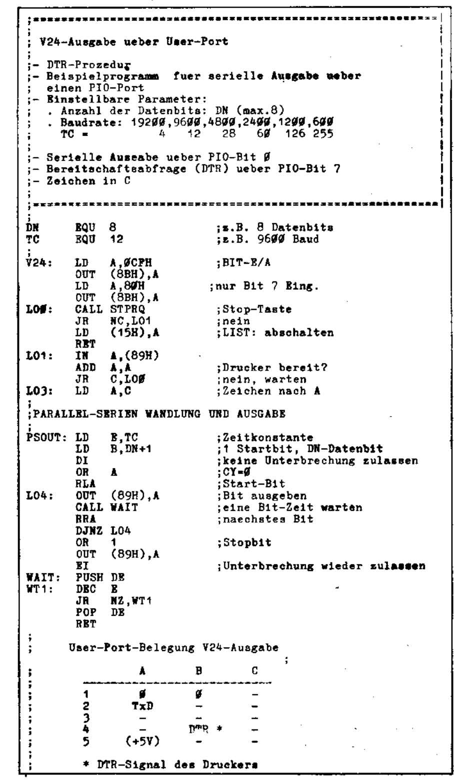
Es wird eine stark abgerüstete, aber weit verbreitete Form dieser weltweit wohl am meisten genutzten Geräteschnittstelle betrachtet. Detailliertere Informationen zu dem Standard können u. a. /2/ entnommen werden.
Computer-Peripheriegeräte mit dieser Schnittstelle arbeiten in der Regel mit einem asynchronen Übertragungsprotokoll nach Bild 3. Dieses Übertragungsprotokoll erlaubt eine große Flexibilität bzgl. bestimmter Parameter:

– Zahl der Datenbits pro Übertragungseinheit (meist 5 . . 8)
– Baudrate (Bit/s): übliche Werte sind 19200, **9600**, 4800, 2400, **1200**, . . .
– keine, gerade oder ungerade Parität
– 1, 1,5 oder 2 STOP-Bits

Peripherieschaltkreise wie der U880-SIO-Schaltkreis ermöglichen die Realisierung dieses Übertragungsprotokolls mit minimalem Aufwand. Sämtliche oben angeführten Parameter sind durch entsprechende Programmierung des Schaltkreises einstellbar. Im allgemeinen sollte für den Aufbau einer seriellen Schnittstelle dieser Art ein solcher (oder ähnlicher) Schaltkreis Verwendung finden. Ein entsprechender Zusatzmodul für den KC 85/1 ist im ZfK Rossendorf entwickelt worden (IFFS und V24).
Wie nachfolgend gezeigt wird, ist in vielen Fällen aber auch ohne Zusatzmodul ein Anschluß solcher Geräte möglich, die mit der sog. DTR-Prozedur arbeiten können. Das bedeutet, daß die Daten zum Gerät seriell übertragen werden und zur Synchronisierung der Übertragungsgeschwindigkeit ein statisches Signal (DTR-Data Terminal Ready) vom Gerät geliefert wird. Der Hardware-Aufwand für den Geräteanschluß (Pegelanpassung) kann unter bestimmten Bedingungen minimal ausfallen (s. u.). Die in Tafel 4 gezeigte Routine übernimmt die serielle Ausgabe eines Bytes über den User-Port entspr. dem in Bild 3 dargestellten Protokoll bei folgenden möglichen Parametern:

– 5 bis 8 Datenbits, ein eventuelles Paritätsbit ist als Datenbit an die Routine zu übergeben.
– Folgende Baudraten sind einstellbar:

Tafel 4 Treiberroutine für V24/DTR



```
;=====================================================
;
; V24-Ausgabe ueber User-Port
;
;- DTR-Prozedur
;- Beispielprogramm  fuer serielle Ausgabe ueber
;  einen PIO-Port
;- Einstellbare Parameter:
;  . Anzahl der Datenbits: DN (max.8)
;  . Baudrate: 192ØØ,96ØØ,48ØØ,24ØØ,12ØØ,6ØØ
;    TC =          4   12   28   6Ø  126 255
;
;- Serielle Ausgabe ueber PIO-Bit Ø
;- Bereitschaftsabfrage (DTR) ueber PIO-Bit 7
;- Zeichen in C
;
;=====================================================
;
DN      EQU  8                 ;z.B. 8 Datenbits
TC      EQU  12                ;z.B. 96ØØ Baud
;
V24:    LD   A,ØCFH            ;BIT-E/A
        OUT  (8BH),A
        LD   A,8ØH            ;nur Bit 7 Eing.
        OUT  (8BH),A
LOØ:    CALL STPRQ             ;Stop-Taste
        JR   NC,L01            ;nein
        LD   (15H),A           ;LIST: abschalten
        RET
L01:    IN   A,(89H)
        ADD  A,A               ;Drucker bereit?
        JR   C,LOØ             ;nein, warten
L03:    LD   A,C               ;Zeichen nach A
;
;PARALLEL-SERIEN WANDLUNG UND AUSGABE
;
PSOUT:  LD   E,TC              ;Zeitkonstante
        LD   B,DN+1            ;1 Startbit, DN-Datenbit
        DI                     ;keine Unterbrechung zulassen
        OR   A                 ;CY=Ø
        RLA                    ;Start-Bit
L04:    OUT  (89H),A           ;Bit ausgeben
        CALL WAIT              ;eine Bit-Zeit warten
        RRA                    ;naechstes Bit
        DJNZ L04
        OR   1                 ;Stopbit
        OUT  (89H),A
        EI                     ;Unterbrechung wieder zulassen
WAIT:   PUSH DE
WT1:    DEC  E
        JR   NZ,WT1
        POP  DE
        RET
;
;     User-Port-Belegung V24-Ausgabe
;
;             A        B        C
;        ---------------------------
;        1    Ø        Ø        -
;        2   TxD       -        -
;        3    -        -        -
;        4    -       DTR *     -
;        5  (+5V)      -        -
;
;        * DTR-Signal des Druckers
```

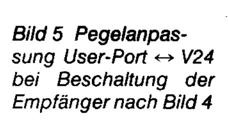
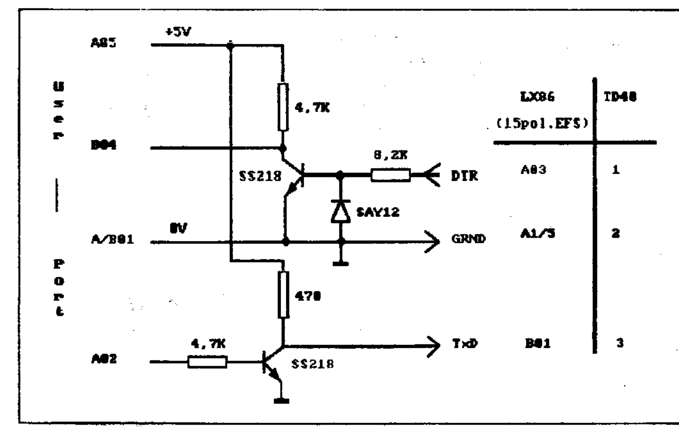


Bild 5 Pegelanpassung User-Port ↔ V24 bei Beschaltung der Empfänger nach Bild 4



Baudrate	Bitzeit/µs	TC (Tafel 4)
19200	52	4
9600	104	12
4800	208	28
2400	416	60
1200	833	126
600	1666	255

Für die Berechnung der Befehlslaufzeiten wurde dabei eine Systemtakt-Frequenz von 2,5 MHz veranschlagt. Niedrigere Baudraten sind leicht durch Einfügen von Dummy-Befehlen in die Warteschleife WT1 (s. Tafel 4) zu erreichen.

– Parität:

Das Paritätsbit wird der Routine als Datenbit übergeben. Die Anzahl der Datenbits (DN in Tafel 4) muß entsprechend definiert werden. Da nur max. 8 Datenbits von der Routine verarbeitet werden, können mit Parität max. sieben Datenbits gesendet werden.

Beispiel: 7 Datenbits und ungerade Parität

```
DN      EQU 8          ;7+1
        .
        .
        LD  A,C        ;ZEICHEN NACH A
        AND 7FH
        RES 7,C
        JP  PO,UPAR    ;UNGERADE PARITAET
        SET 7,C
UPAR:   CALL V24       ;AUFRUF DER ROUTINE
                        NACH TAFEL 2
        .
        .
```

Ein Zwangsausgang über die STOP-Taste wurde wiederum vorgesehen.

Man beachte, daß die gezeigte Routine das unterste (d. h. physikalische) Übertragungsprotokoll realisiert. Druckerspezifische Anpassungen (Sonderzeichen, Steuerzeichen o. ä.) sollten an dieser Stelle nicht erfolgen, sondern – sofern überhaupt erforderlich – von einem übergeordneten Treiberprogramm ausgeführt werden. Die dargestellte Routine ist allerdings direkt für den Betrieb eines TD40 (K6303) oder auch des EPSON LX86 (PC-1715-Variante) am KC 85/1 geeignet.

Den notwendigen Pegelanpassungen muß hier besondere Aufmerksamkeit gewidmet werden. Standardmäßig finden beim RS232C die Treiberschaltkreise 75150, 75188 o. ä. und die Empfängerschaltkreise 75154, 75189 o. ä. Verwendung. Damit ist eine normgerechte Pegelanpassung durchführbar, allerdings werden für die Treiberschaltkreise +/−12 V bzw. +/−9 V benötigt, die am User-Port leider nicht zur Verfügung stehen. Hier sollte im konkreten Fall geprüft werden, wie die Empfängerschaltkreise im Gerät beschaltet sind. Die angegebenen Typen gestatten nämlich eine Verschiebung der Triggerschwelle. Bild 4 zeigt eine Beschaltung der Empfängerschaltkreise, die eine sichere Ansteuerung mit einem Signalpegel von 0 bzw. +5 V erlaubt (z. B. im LX86 und K6303 gegeben). Damit entfällt die Notwendigkeit einer negativen Spannung, und eine Pegelanpassung nach Bild 6 ist der gesamte Hardware-Aufwand für den Druckeranschluß. Die Kabellänge sollte dann allerdings 2 bis 3 m nicht überschreiten.

2.3. CENTRONICS

Das CENTRONICS-Interface ist eine bitparallele bidirektionale Datenschnittstelle, die insbesondere als Druckerinterface z. Z. weltweit starke Verbreitung findet. Alle modernen Drucker für Heim- und Personalcomputer bieten in der Regel auch eine Ausrüstungsvariante für diese Schnittstelle.

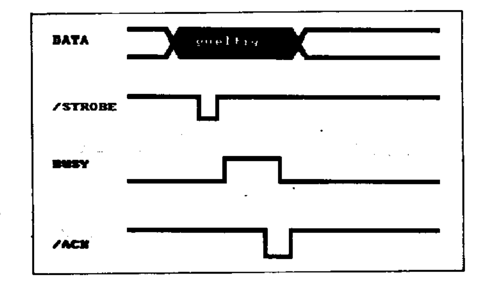


Bild 6 CENTRONICS-Signalverhalten

Bild 6 zeigt das Signalverhalten. Da die Pegel TTL-Pegel sind, eignet sich dieses Interface besonders gut für den Betrieb am User-Port des KC 85/1. Bestimmte Einschränkungen müssen allerdings wegen der begrenzten Zahl der Datenleitungen wieder getroffen werden. Will man gänzlich ohne Zusatzhardware auskommen, so bietet sich eine Zuordnung der Signale des User-Ports zu den CENTRONICS-Signalen in folgender Form an:

User-Port	CENTRONICS-Druckerschnittstelle
PIO-0	DATA-1
PIO-1	DATA-2
PIO-2	DATA-3
PIO-3	DATA-4
PIO-4	DATA-5
PIO-5	DATA-6
PIO-6	DATA-7
PIO-7	/STROBE
CTC-CNT/TRG	/ACK

Es ist zu erkennen, daß bei dieser Zuordnung das PIO-Bit 7 für die Erzeugung des CENTRONICS-/STROBE-Signals verwendet wird und somit nur 7 Datenbit übertragen werden können. Die meisten modernen Matrix-Drucker ermöglichen auch bei 7 Datenbits alle verfügbaren Druckmodi (einschließlich Grafik). Bei geschickter Ausnutzung des Signalverhaltens der RDY-/STRB-Leitungen kann das RDY-Signal auch als CENTRONICS-/STROBE-Signal verwendet werden. Die damit erreichbare Länge des /STROBE-Signals liegt dabei aber immer unter dem normalerweise erforderlichen Wert von mindestens 1 µs. Bei einigen Druckern funktioniert das zuverlässig (Schneider NLQ 401, Seikosha GP100), andere Drucker haben damit Probleme (K6313). Die oben angeführte Zuordnung sichert das Standard-Signalverhalten und ist ohne jede Zusatzhardware einsetzbar.

Auf eine Auswertung der Papierendemeldung vom Drucker (PE) muß aus den genannten Gründen auch hier verzichtet werden.

Tafel 3 zeigt die entsprechende Treiberroutine für den Anschluß eines CENTRONICS-Druckers an den User-Port des KC 85/1.

3. Einbinden von Gerätetreibern in das Betriebssystem (OS) des KC 85/1

Das OS des KC 85/1 unterstützt leistungsfähig das Einbinden neuer Gerätetreiberroutinen für die sog. zeichenorientierten logischen Ein-/Ausgabekanäle. Vorbild für das im folgenden kurz erläuterte Konzept ist das Betriebssystem CP/M.

3.1. I/O-Byte-Konzept

Das OS-KC 85/1 verwaltet eine Tabelle mit 4 mal 4 Plätzen für Adressen von Gerätetreiberroutinen (Tafel 5). Jeweils eine Zeile dieser Tabelle wird dabei einem der 4 logischen E/A-Kanäle

CONST: (Console)
READER: (zusätzlicher Eingabekanal)
PUNCH: (zusätzlicher Ausgabekanal)
List: (log. Kanal für Drucker)

zugeordnet, d. h. für jeden log. E/A-Kanal können 4 Treiberroutinen gleichzeitig resident sein. Welche dabei im gegebenen Moment ausgewählt wird, bestimmt die Belegung des I/O-Bytes (HS-Adr. 0004, im CP/M üblicherweise Adr. 0003).

Das OS selbst beinhaltet zwei residente Treiberroutinen:

CRT Consoltreiber
BAT Batch-Treiber

CRT ist der Treiber für die Standard-Console, d. h. für das Fernsehgerät als Sichtgerät und die eingebaute Tastatur.

BAT ist ein Treiber, der das log. Gerät Console (CONST:) auf die Kanäle READER: (Eingabe) und LIST: (Ausgabe) aufteilt, also nur verwendbar ist, falls auch für READER: und LIST: entsprechende Treiber eingebunden wurden.

Alle anderen Plätze der Tabelle sind nach einem Kaltstart des OS (RESET oder POWER ON) mit dem Wert 0FFFFH belegt und zeigen dem OS damit einen nicht existenten Treiber an. Sollen eigene Treiberroutinen – wie z. B. die in Tafel 1 bis 4 – eingebracht werden, so kann das durch direktes Eintragen der Treiberadresse in den entsprechenden Tabellen-

```
          \ phys.Ger.
log.Ger.|   Ø    |   1    |   2    |   3    |                   Bit
--------|--------|--------|--------|--------|          ------
        | EFC9   | EFCB   | EFCD   | EFCF   |          |    |   Ø
CONST:  |        |        |        |        |   <==    |Ø..3|
        | (TTY)  | (CRT)  | (BAT)  | (UC)   |          |    |   1
--------|--------|--------|--------|--------|          |----|
        | EFD1   | EFD3   | EFD5   | EFD7   |          |    |   2
READER: |        |        |        |        |   <==    |Ø..3|
        | (TTY)  | (RDR)  | (UR1)  | (UR2)  |          |    |   3
--------|--------|--------|--------|--------|          |----|
        | EFD9   | EFDB   | EFDD   | EFDF   |          |    |   4
PUNCH:  |        |        |        |        |   <==    |Ø..3|
        | (TTY)  | (PUN)  | (UP1)  | (UP2)  |          |    |   5
--------|--------|--------|--------|--------|          |----|
        | EFE1   | EFE3   | EFE5   | EFE7   |          |    |   6
LIST:   |        |        |        |        |   <==    |Ø..3|
        | (TTY)  | (CRT)  | (LST)  | (UL1)  |          |    |   7
--------------------------------------------          ------

Adresstabelle der physischen Treiberroutinen         I/O-Byte
                                                     (HS-Adr.4)
```

Tafel 5 I/O-Byte und Zuordnungstabelle im OS-KC 85/1

```
   Beispiel fuer das Einbinden einer Druckertreiberroutine
   ========================================================


        ORG     hhØØH            ;Low-Byte = Ø!!
        jp      ILX86            ;Init-Routine
STR:    defb    'LX86            ;Name=LX86
        defw    Ø                ;Ende-Kennzeichen
;
;
ILX86:  ld      a,Øcfh           ;Port
        out     (8BH),a          ;initialisieren
        ld      a,8ØH            ;nur Bit 7 Eingabe
        out     (8BH),a
        .
        .                        ;hier evtl.
        .                        ;Druckerinitialisierung
        .                        ;etc. durchfuehren
        .
        ld      hl,6*256+2       ;LIST:, phys.Ger.2
;                                 phys.Ger. Ø nicht sinnvoll,
;                                 da ASGN CONST:=LX86 moeglich
;                                 waere!
        ld      bc,V24           ;Treiber nach Tafel 4
        ld      de,STR           ;Adr. des symb. Namens
;
;       ex      (sp),hl          ;dieser Teil
;       inc     hl               ;ist nur bei OS Version 1.1
;       inc     hl               ;noetig (wegen eines Fehlers
;       ex      (sp),hl          ;in der ASGN-Routine)
;
        or      a                ;CY=Ø, wenn kein Fehler
        ret
;
V24:    inc     a                ;Drucker-Initialisierung?
        ret     z                ;z.B. ignorieren
        dec     a                ;Drucker-Status?
        jr      nz,loØ           ;nein
;
        in      A,(89H)          ;Status
        or      7FH              ;z.B. so
        cpl                      ;zurueckgeben
        ret
loØ:    weiter s. Tafel 4
        .
        .
        .
```

Tafel 6 Einbinden eines Druckertreibers in das OS-KC 85/1

Tafel 7 HEX-Dump des Treiberpakets

```
0300  C3 51 03 53 49 46 45 20 .Q.SIFE      0308  20 20 20 00 C3 5B 03 53    ..[.S
0310  49 46 41 20 20 20 20 00 IFA    .     0318  C3 64 03 43 45 4E 54 52 .d.CENTR
0320  20 20 20 00 C3 6D 03 4C    ..m.L     0328  58 38 36 20 20 20 20 00 X86    .
0330  C3 74 03 54 44 34 30 20 .t.TD40      0338  20 20 20 00 C3 49 03 42    ..I.B
0340  45 45 50 20 20 20 20 00 EEP    .     0348  00 01 59 04 21 03 00 18 ..Y.!...
0350  44 01 3E 04 21 02 02 16 D.>.!...     0358  7F 18 2E 01 06 04 2E 03 ........
0360  16 00 18 1C 01 DC 03 2E ........     0368  01 16 00 18 13 01 A8 03 ........
0370  3E 0C 18 05 01 A8 03 3E >......>     0378  7E 32 C3 03 2E 02 16 FE ~2......
0380  3A 33 00 67 D6 04 3E 01 :3.g..>.     0388  D8 3E CF D3 8B 7A D3 8B .>...z..
0390  2F D3 89 16 03 B7 E3 23 /......#     0398  23 E3 C9 CD 33 FD FB D6 #...3...
03A0  03 B7 C0 32 25 00 37 C9 ...2%.7.     03A8  3C C8 3D 20 06 DB 89 F6 <.= ....
03B0  7F 2F C9 CD 9B 03 30 04 ./....0.     03B8  32 15 00 C9 DB 89 87 38 2......8
03C0  F2 79 1E 7E 06 09 F3 B7 .y.~....     03C8  17 D3 89 CD D6 03 1F 10 ........
03D0  F8 F6 01 D3 89 FB D5 1D ........     03D8  20 FD D1 C9 3C C8 3D 2F  ...<.=/
03E0  C8 3E 57 D3 81 D3 81 79 .>W....y     03E8  F6 80 FE FF 20 02 00 00 .... ...
03F0  D3 89 E6 7F D3 89 F6 80 ........     03F8  D3 89 CD 9B 03 D8 DB 81 ........
0400  FE 57 28 F6 AF C9 3E 47 .W(...>G     0408  D3 81 D3 81 79 2F CB FF ....y/..
0410  D3 89 00 00 CB BF D3 89 ........     0418  CD 9B 03 30 04 2F D3 89 ...0./..
0420  C9 DB 81 FE 47 28 F1 3E ....G(.>     0428  57 D3 81 D3 81 3E FF D3 W....>..
0430  89 CD 9B 03 38 E7 DB 81 ....8...     0438  FE 57 28 F5 B7 C9 3E 47 .W(...>G
0440  D3 81 D3 81 D3 89 CD 9B ........     0448  03 38 D2 DB 81 FE 47 28 .8....G(
0450  F5 DB 89 2F D3 89 E6 7F .../....     0458  C9 CD F1 F8 F5 3A 2F 00 .....:/.
0460  3D CC 66 04 F1 C9 C5 06 =.f.....     0468  00 0E 14 3E 07 D3 80 3E ...>...>
0470  96 D3 80 DB 88 CB FF D3 ........     0478  88 10 FE 0D 20 FB CB BF .... ...
0480  D3 88 3E 03 D3 80 C1 C9 ..>.....     0488  0E 09 11 03 05 CD 05 00 ........
0490  2A 36 00 01 88 01 B7 ED *6......     0498  42 2E 00 E5 2B 22 36 00 B...+"6.
04A0  C5 E5 06 10 11 00 00 4B .......K     04A8  29 7B 8B 27 5F 7A 8A 27 ){.'_z.'
04B0  57 79 89 4F 10 F2 79 B7 Wy.O..y.     04B8  EB 28 03 CD F0 04 CD EB .(......
04C0  04 11 1D 05 0E 09 CD 05 ........     04C8  00 E1 C1 23 EB 21 00 03 ...#.!..
04D0  ED B0 E1 DD 21 20 05 16 ....! ..     04D8  03 5C 06 00 DD 4E 00 0C .\...N..
04E0  0D C8 09 7E 92 83 77 DD ...~..w.     04E8  23 18 F1 7C CD F0 04 7D #..|...}
04F0  F5 0F 0F 0F 0F CD F9 04 ........     04F8  F1 E6 0F C6 30 0E 02 5F ....0.._
0500  C3 05 00 0D 0A 54 6F 70 .....Top     0508  20 6F 66 20 52 41 4D 20  of RAM
0510  6E 6F 77 20 28 64 65 63 now (dec     0518  2E 29 3A 20 00 0D 0A 00 .): ....
0520  02 0C 0C 0C 0C 0C 0D 08 ........     0528  0A 09 09 07 05 19 21 18 ......!.
0530  2F 1E 19 15 1B 00 00 00 /.......
```

platz und zugehöriges Stellen des I/O-Bytes erfolgen. Aber auch hierfür bietet das OS ein leistungsfähiges Hilfsmittel – das ASGN (Assign)-Kommando – an. Diese Form des Einbindens von E/A-Treibern sollte i. a. bevorzugt werden, da die konkreten Anordnungen von Zuordnungstabelle (Tafel 5) und I/O-Byte im Speicher damit für den Nutzer transparent bleiben (d. h., evtl. Veränderungen wirken sich nicht auf das Anwenderprogramm aus).

3.2. ASGN-Kommando

ASGN ist ein residentes Kommando des OS (von insgesamt 3). Die Aufrufform ist

ASGN[log. Gerät = phys. Gerät]

ASGN ohne Parameter gibt die aktuelle Belegung auf dem Bildschirm aus, z. B.

CONST: = CRT
READER: =
PUNCH: =
LIST: =

sofern noch keine zusätzlichen Treiber eingebunden worden sind. Die möglichen logischen Geräte wurden bereits erläutert, als phys. Geräte sind **CRT, BAT** oder Namen eigener Treiberroutinen möglich (max. 8 Buchstaben oder Zahlen).

Beispiel: **ASGN LIST: = LX86**

Wird eine solche Anweisung gegeben, so vollzieht sich im OS ein umfangreicher Mechanismus:
Zuerst wird eine Treiberroutine namens "LX86" im Speicher (von oben nach unten) entsprechend den Aufrufregeln für transiente Kommandos gesucht:
– Suche auf allen Adressen mit Low-Byte = 0 nach einer Bytefolge der Form: C3 ll hh (= JP hhllH)
– Die folgenden 8 Bytes werden mit der angegebenen Zeichenkette (hier "LX86") verglichen.

Diese Tabelle wird mit einem Nullbyte abgeschlossen und kann mit weiteren Strukturen dieser Form fortgesetzt werden. Zwei aufeinander folgende Nullbytes beenden diese Sprungtabelle (vgl. Beispiel Tafel 6).
Ist die Suche erfolgreich, so wird die Steuerung zur Adresse hhllH übergeben (Initialisierungsroutine des Treibers), andernfalls wird angenommen, daß eine Treiberroutine vom Band nachgeladen werden soll, und es erfolgt die Ausschrift

start tape.

Nach dem Einlesen wird die Routine an der im FCB (1. Block) spezifizierten Startadresse gestartet. Im allgemeinen sollte die Treiberroutine an das aktuelle Speicherende (RAM) plaziert werden (über Zelle 36 H zu ermitteln). Dem OS müssen nun noch einige Informationen übermittelt werden:
– Welchem log. Gerät darf die Treiberroutine zugeordnet werden?
– Die Startadresse des Treiberteils
– Auf welchem Platz der Zuordnungstabelle (Tafel 5) soll der Eintrag erfolgen (physisches Gerät 0...3)?
– Die Adresse des symbolischen Gerätenamens (8-Byte-String).

Zu beachten ist, daß physische Geräte mit der Nr. 0 jedem log. E/A-Kanal zugeordnet werden können (also auch der Console) und entsprechend aufgebaut sein müssen.
Ein physisches Gerät mit der Nr. 1 kann auch immer dem LIST-Kanal zugeordnet werden (Beispiel: residenter CRT-Treiber).
Die Parameterübergabe erfolgt über die folgenden Register:

H: log. Gerätenr. ⇒ kennzeichnet den log. Kanal, dem die Treiberroutine zugeordnet werden kann.
  Ø – CONST:
  2 – READER:
  4 – PUNCH:
  6 – LIST:
L: phys. Gerät (0, ..., 3)
BC: Adresse der Treiberroutine
DE: Adresse der Zeichenkette (symbolischer Gerätename)

Tafel 6 gibt ein Beispiel an, wie die Treiberroutine nach Tafel 4 (V24/DTR) als Druckertreiber mit der Bezeichnung "LX86" in das OS eingebunden werden kann. Mit der Anweisung

ASGN LIST: = LX86

wird die Routine V24 (Tafel 4) eingebunden und das I/O-Byte entsprechend gestellt. Es erfolgt die Ausschrift

CONST: = CRT
READER:
PUNCH:
LIST: = LX86.

Der Drucker kann jetzt mit CTRL/P zum Bildschirm parallel geschaltet werden (auch im BASIC) oder über den LIST-Kanal (Ruf-Nr. 5) direkt angesprochen werden.

3.3. Treiberpaket

Tafel 7 zeigt den HEX-Dump eines kompletten Treiberpakets, das u. a. die vorgestellten Treiber enthält.

Das Programm wird normal (nicht mit ASGN) geladen. Es enthält einen Mechanismus, der das eigentliche Treiberpaket an das aktuelle RAM-Ende verschiebt und eine entsprechende Adreßmodifikation vornimmt. Das Programm ist damit für jede RAM-Konfiguration geeignet und muß nicht speziell angepaßt werden. Die aktuelle obere Speichergrenze wird für BASIC-Anwender dezimal ausgegeben. Dieser Wert sollte nach Start des BASIC-Interpreters als obere Grenze eingegeben werden, um das Treiberpaket vor Überschreiben zu schützen.
Die gewünschten Treiber **müssen** vorher mit

der ASGN-Anweisung aktiviert werden. Dabei sind die folgenden Zuordnungen möglich:

log. Gerät	physische Gerätenummer 0	1	2	3
CONST:		(CRT)	(BAT)	BEEP
READER:			SIFE	
PUNCH:		CENTR	LX86 TD40	SIFA
LIST:		CENTR	LX86 TD40	SIFA

BEEP: Erzeugt einen „sauberen" (im Gegensatz zu CTRL/Q) Tastatur-Quittungston, der die Eingabesicherheit der Tastatur verbessert.
SIFE: SIF1000-Eingabe, 7 Datenbits parallel, siehe Tafel 2
SIFA: SIF1000-Ausgabe, 7 Datenbits parallel, siehe Tafel 1
CENTR: 7-Bit-CENTRONICS-Druckerinterface, siehe Tafel 3
TD4Ø: V24/DTR, 8 Bit, keine Parität, 1200 Baud, siehe Tafel 4
LX86: V24/DTR, 8 Bit, keine Parität, 9600 Baud, siehe Tafel 4
Das Programm kann z. B. mit dem von Robotron vertriebenen Zusatzmonitor ZM eingegeben werden (S-Kommando). Mit dem P(unch)-Kommando kann es dann (nach vorheriger Zuweisung AR=T) auf Band gebracht werden:
P300,537,488.

Man beachte die Startadresse (488H).
Der erwähnte Zusatzmonitor enthält bereits einige der hier vorgestellten Treiberroutinen (SIFA/E, TD40, BEEP). Zu beachten ist, daß es bei den Treibern SIFA und TD40 des ZM u. U. Probleme mit der CTRL/P-Funktion (Hardcopy) geben kann (1. Version des ZM und Version 1.2 des OS-KC85/1). Deshalb sei an dieser Stelle auf die erforderlichen Korrekturen für eine saubere Arbeitsweise der ZM-Treiber hingewiesen:

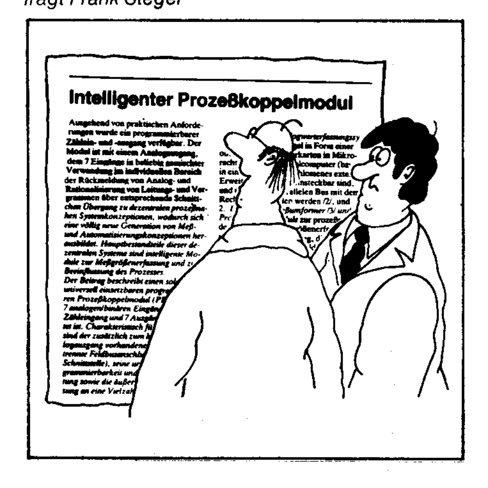
Treiber	relative Adresse	alt Befehl	HEX ⇒	neu Befehl	HEX
SIFA	62H	ld a,e	7B	ld a,c	79H
TD40	C3H	res 7,e	CB BB	res 7,c	CB B9
	E0H	sub e	93	sub c	91
	E8H	cp e	BB	cp c	B9
	FCH	ld a,e	7B	ld a,c	79
	102H	ld c,154	0E 9A	ld e,154	1E 9A
	117H	ld b,c	41	ld b,e	43

Der ZM nutzt **nicht** die ASGN-Anweisung. Mit RESET oder POWER ON wird die Zuordnungstabelle (Tafel 5) mit den entsprechenden Adressen geladen, da alle Treiber resident sind. Die Auswahl der ZM-Treiber muß deshalb mit dem ZM-Kommando A(ssign) erfolgen (Stellen des I/O-Bytes).

✉ KONTAKT ☎
Zentralinstitut für Kernforschung
der AdW der DDR Rossendorf,
PSF 19, Dresden, 8051, Tel.: 591 2377

Literatur
/1/ Mikroprozessorsystem der 2. Leistungsklasse, Beschreibungen der Bausteine CPU 880, PIO, SIO, CTC und DMA. VEB Mikroelektronik „Karl Marx" Erfurt
/2/ Wilhelm, C.: Datenübertragung. Berlin: Militärverlag der DDR, 1976
/3/ Schwarzenberg, F.; Wobst, R.: Anschluß der Schreibmaschine S 6005 an KC 85/1. Mikroprozessortechnik, Berlin 1 (1987) 10, S. 315
/4/ Dokumentation des Betriebssystems OS-KC 85/1, VEB Robotron Vertrieb Berlin

„Gibt es auch einen dummen Prozeßkoppelmodul?"
fragt Frank Steger



Anschluß der Schreibmaschine S 6005 an KC 85/1

Dr. Frank Schwarzenberg,
Dr. Reinhard Wobst,
Dresden

Hardware

Die Lösung wurde speziell für die Nutzung der S 6005 als Ein- und Ausgabegerät für den Kleincomputer 85/1 erarbeitet, ist aber ohne Hardwareänderungen für jeden anderen Rechner mit einem verfügbaren freiprogrammierbaren 8-Bit-Port geeignet. Die folgenden Randbedingungen wurden bei der Entwicklung beachtet:

- kein Eingriff in die S 6005
- minimaler Hardwareaufwand
- Nachnutzbarkeit
- Ansteuerung über programmierbaren 8-Bit-Port
- Schreibmaschine als Drucker und Tastatur verwendbar.

Da kein Eingriff in Elektronik und Firmware der Schreibmaschine erfolgen sollte, wurde als Schnittstelle die Tastaturmatrix der S 6005 gewählt. Das ermöglicht außerdem minimalen Hardwareaufwand, begrenzt aber andererseits die maximal mögliche Druckgeschwindigkeit (keine Rückmeldung der Schreibmaschine über Druckausführung).

Die Tastaturmatrix ist bei der S 6005 in 9 Spalten und 8 Zeilen aufgeteilt. In den Schnittpunkten der Matrix sind die Tasten angeordnet. Bei der Verwendung als Drucker muß das Schließen der Tasten vom Rechner simuliert werden, bei der Nutzung als Tastatur muß vom Rechner die Tastaturmatrix erregt und abgefragt werden.

Eine direkte Ansteuerung der Matrix erfordert 9 + 8 = 17 Bit plus ein Bit zum Verhindern der Tastaturabfrage durch den Mikroprozessor der Schreibmaschine. Da am KC 85/1 dem Nutzer nur ein PIO-Port zur Verfügung steht, wurde eine Lösung entsprechend Bild 1 gewählt:
Zwei bidirektionale Multiplexer/Demultiplexer mit interner Decodierung und ein 4-Bit-Latch werden benötigt. Durch Codieren der Zeilen- (3 auf 8) und Spaltenauswahl (4 auf 9) sowie zusätzliches Zwischenspeichern der Zeileninformation werden nur 4 Bit für die Zeilen- und Spaltenauswahl benötigt.

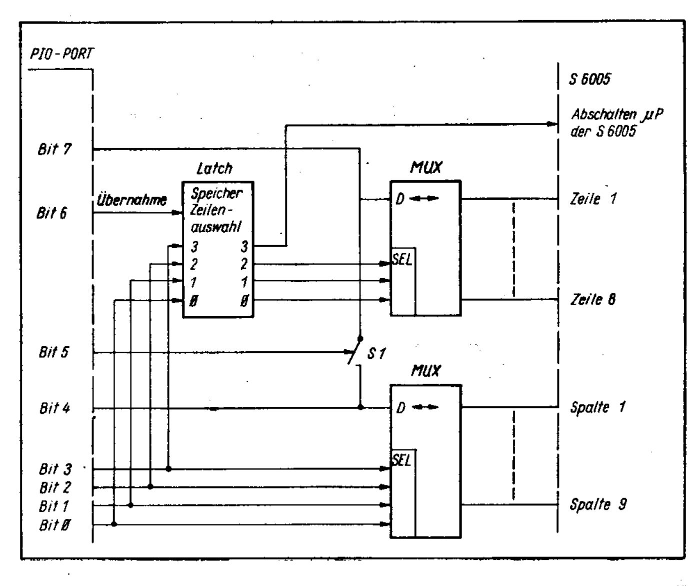


Bild 1 Prinzipschaltung

Beim Drucken wird über den Schalter S1 direkte Verbindung zwischen dem Spalten- und Zeilensignal für eine bestimmte (durch die Tastaturentprellung festgelegte) Zeit hergestellt und damit das Betätigen einer Taste simuliert.
Es können nicht gleichzeitig zwei Tastendrücke simuliert werden, was aber praktisch nicht stört (s. 2.).
Für die Tastaturabfrage ist Schalter S1 geöffnet, über Bit 3 der Zeileninformation wird der interne Mikroprozessor der Schreibmaschine abgeschaltet. Die Matrix kann nun über Bit 7 zeilenweise erregt und eine eventuelle Tastenbetätigung über Bit 4 spaltenweise abgefragt werden.

Software

Das Programm ist 1,5 KByte lang und umfaßt folgende Teile:
- Drucker und Tastaturtreiber
- Zuweisung des Druckertreibers und Hin- und Herschalten zwischen S6005 und KC85/1-Tastatur durch einfache Kommandos
- einfaches Druckprogramm zum Ausdrukken von Textfiles.

Druckertreiber

Die einzelnen ASCII-Zeichen werden teils mit mehreren Tastendrücken, die nacheinander simuliert werden, erzeugt – z.B. "A" durch SHIFT LOCK, "a" und SHIFT. Dieses scheinbar umständliche Vorgehen ist notwendig, da ja die Hardware nur einen „Tastendruck" auf einmal zuläßt. Manche Zeichen werden auch zusammengesetzt, z.B. die geöffnete eckige Klammer – auf dem Typenrad nicht vorhanden – aus "_", Rücktaste, "0", zweimal "hoch", "_" und zweimal "runter":[. Auch der auf dem Typenrad vorhandene senkrechte Strich muß simuliert werden, da KB-Taste und Taste "4" nicht gleichzeitig gedrückt werden können:¦. Allerdings ist das das einzige wichtige Zeichen, bei dem sich die einfache Hardware beschränkend auswirkt.
Die Codierung der Zeichen wird aus einer leicht änderbaren Tabelle entnommen.
Umlaute und andere im ASCII-Code nicht enthaltene, aber auf der Tastatur vorhandene Zeichen werden als Steuerzeichen (Code 32) angesprochen. Cursorsteuerzeichen behalten ihre Funktion.
Folgt auf „return" (Code 13) ein „linefeed" (Code 10), so wird das „linefeed" als Taste „Zeilenwechsel" ausgegeben. Es kann also bei Bedarf auch 1-, 1½- oder 2zeilig gedruckt werden, nur durch Verstellen des Zeilenschalters.
Hauptproblem bei Programmierung des Treibers war, daß von der Schreibmaschine keine Rückmeldung nach Druck eines Zeichens erfolgt. Es ist also notwendig, bei jedem Zeichen die maximal mögliche Zeit zu warten.
Die Zeit für den Wagenrücklauf wurde mit guter Näherung als lineare Funktion der Zeichenzahl in der Zeile berechnet.
Die Schreibgeschwindigkeit beträgt etwa 6,5 Zeichen/s (ohne Wagenrücklauf). Der Zeitverlust gegenüber dem optimalen Schreibtempo kann bis Faktor 2 betragen. Andererseits sollte man nicht vergessen, daß die vorgestellte Lösung keinen professionellen Drucker ersetzen kann, sondern eher für den „Hausgebrauch" gedacht ist. Und gerade deswegen ist wichtiger, daß jede Schreibmaschine Lärm erzeugt, der beim maximalen Tempo schon recht beträchtlich ist.

Tastaturtreiber

Bei jeder Tastaturabfrage müssen alle Kontakte der Matrix abgefragt werden. Die Abfragezeit beträgt etwa 4ms, spielt also kaum eine Rolle. Der Treiber benutzt den CTC-Kanal 1.
Gegenüber der KC 85/1-Tastatur sind folgende Veränderungen wichtig:
- Die Wiederholfunktion ist schneller (13 Zeichen/s).
- Alle ASCII-Zeichen sind darstellbar.
- Es gibt Sondertasten für CONTROL, GRAPHIC, STOP, PAUSE, INS, DEL, ESC und COLOR sowie für die Cursorbewegungen.
- Folgende Arten der Groß- und Kleinschreibung sind möglich:
- Schreibmaschinenmode: Buchstaben normal klein geschrieben, bei SHIFT groß.
- KC85/1 – Mode: Buchstaben standardmäßig groß, bei SHIFT klein.
- SHIFT LOCK wirkt stets wie SHIFT und wird durch erneutes Drücken von SHIFT LOCK wieder abgeschaltet. Die entsprechende LED zeigt an, ob SHIFT LOCK eingeschaltet ist.
Die Taste hat keine Verzögerung wie beim KC85/1.
Solange zusätzlich SHIFT gedrückt ist, hebt es SHIFT LOCK auf.
- Grafiktaste ist die Sperrtaste. Der Grafikmode wird durch sie wie üblich abwechselnd ein- und ausgeschaltet und durch beide LEDs auf S6005 und KC85/1 angezeigt. Auch die Grafiktaste arbeitet verzögerungsfrei.
- Grafikzeichen werden anders als beim KC85/1 eingegeben: Zum Code des eingegebenen Zeichens wird im Grafikmode einfach 128 addiert.
- Eine bestimmte Tastenkombination löst Systemneustart, d.h. Sprung zu F000H, aus.
- Es kann eine Hardcopy des Bildschirms erzeugt werden.
- Nicht darstellbar sind die Zeichen 127 und 255.

Zuweisungsroutinen

Um nicht jedesmal ein ASGN-Kommando geben zu müssen, werden die entsprechenden Kommandozeilen dem Betriebssystem durch spezielle Programme „untergeschoben". Zur Druckerzuweisung ist so z.B. nur „AL" zu geben.

Druckprogramm

Das Druckprogramm verarbeitet eine Kommandozeile mit Standardannahmen und leistet folgendes:
- Ausdruck eines im Speicher stehenden Textes ab eingegebener Adresse
- Wahlweise automatische Umwandlung von ae AE, oe usw. in Umlaute,
- Ausgabe von nicht-ASCII-Zeichen wie μ, ß, ^{2} usw. und Steuerzeichen durch spezielle Zeichenkombinationen,
- Automatische Seitennumerierung rechts unten, ab gewünschter Anfangsnummer. Das Programm unterbricht den Druck nach jeder Seite,
- Wahlweises Einstellen der Zeilenzahl pro Seite,
- Seitenvorschub durch Sonderzeichen,
- Möglichkeit, jederzeit das Drucken zu unter- oder abzubrechen bzw. fortzusetzen.

Das gesamte Programmpaket wird seit Oktober 86 intensiv genutzt und hat sich bisher ausgezeichnet bewährt.

Z.Z. beinhaltet die Nachnutzung
- einen Schaltplan,
- Software für den KC85/1 entsprechend der Kundenkonfiguration,
- Dokumentation,
- kostenlose Fehlerbeseitigung sowie
- Nachnutzungsmöglichkeit für Weiterentwicklung.

✉ KONTAKT ☎
Zentralinstitut für Festkörperphysik und Werkstoffforschung der AdW der DDR, Dr. R. Wobst, Postfach, Dresden, 8027, Tel. 4635555

Termine

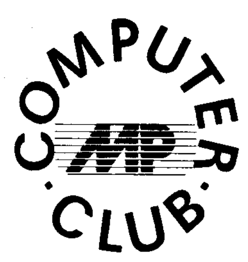
6. Fachtagung Mikroelektronik 88

mit internationaler Beteiligung aus den RGW-Ländern

WER? Fachverband Elektrotechnik in der Kammer der Technik, wissenschaftliche Sektion Bauelemente der Elektrotechnik/Elektronik in Abstimmung mit der Föderation der sozialistischen Länder – FeNTO
WANN? 7. bis 9. Dezember 1988
WO? Dresden
WAS?
1. Einschätzung des internationalen Standes und der weiteren Tendenzen der Entwicklung und Anwendung der Mikroelektronik
2. Informationen zur Entwicklung und Anwendung moderner elektronischer Bauelemente
3. Verfügbare Kleincomputer für Rationalisierungsaufgaben
4. Schlüsseltechnologie Mikroelektronik – Basis für komplexe und automatisierte flexible Automatisierungsaufgaben
5. Posterdiskussion und Angebotsmesse für nachnutzbare mikroelektronische Lösungen, insbesondere für Rationalisierungsaufgaben

WIE? Vortragsmeldungen/Poster mit Titel und Inhaltsangabe (max. 10–15 Zeilen) bis zum 15. Januar 1988 – ebenso wie Teilnahmemeldungen schriftlich – an:
Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Clara-Zetkin-Str. 115/117, Berlin, 1086

Hoppe



Arbeitsspeicher-umordnung für KC 85/2

Der für den Stack und die Interrupttabelle reservierte Arbeitsspeicher belegt im KC85/2 den Bereich ab 200H abwärts. Das Betriebssystem erlaubt es jedoch, ihn an eine andere Stelle im RAM anzuordnen. Dabei werden von CAOS folgende Bedingungen eingestellt:
Das IX-Register zeigt auf Arbeitszellen zur Steuerung der Ein- und Ausgabe. Bei einer Verlagerung des Arbeitsbereiches bleibt der niederwertige Teil erhalten (ab xxF0).
Die Interrupttabelle liegt zwischen xxD4 und xxEC.
Der Stack wird unter die Interrupttabelle gelegt. Bei Systemstart und Reset ist xx auf 01 eingestellt. Durch das folgende Programm wird der höherwertige Teil des Arbeitsspeichers auf einen neuen Wert gesetzt. Es wird mit

WORKRAM adr

gestartet, wobei „adr" der höherwertige Teil der neuen Anfangsadresse des Arbeitsspeichers ist. Das Originalprogramm liegt im Bereich 200H–21FH.

7F	7F	57	4F	52	4B	52	41
4D	01	AF	B4	C0	F3	7D	D1
21	00	00	39	67	F9	22	AE
B7	D5	CD	03	F0	31	FB	C9

Der Anwender muß selbst sichern, daß keine Kollision mit bereits geladenen Programmen auftritt bzw. nachfolgend gestartete Programme (BASIC!) den Arbeitsbereich nicht überschreiben.
Das Programm ist ebenfalls auf dem KC85/3 lauffähig.

M. Lennartz

Behandlung externer Interruptquellen bei KC 85/1 und KC 87

Bei der Behandlung externer Interruptquellen (Interruptquelle ist über Systembus im Modulschacht an den KC angeschlossen) sind bei den robotron-Kleincomputern die folgenden Hinweise zu beachten.
Der Modul mit der externen Interruptquelle muß auf dem ersten Steckplatz (von der Tastatur aus gesehen) gesteckt sein. Damit wird diese Interruptquelle an die bestehende Interruptprioritätenkette angeschlossen.
Die peripheren Systemelemente des Systems U880 erkennen durch interne Dekodierung des Befehls RETI, daß ihre Interruptbearbeitungsroutine (ISR) beendet ist, und können somit ihren Bearbeitungszustand zurücksetzen.
Bei externem Interrupt muß der Befehl RETI als Abschluß der ISR auf einer Speicheradresse zwischen 4000H und 0EFFFH stehen, da sonst das den externen Interrupt auslösende Systemelement diesen RETI-Befehl nicht dekodieren kann. Das ist in der Schaltungskonzeption begründet, bei der die Datenbustreiber, welche den Datenbus vom bzw. zum Modulschacht treiben, nur dann aktiv sind, wenn eine Speicheradresse zwischen 4000H und 0EFFFH angesprochen wird.
Zur Erfüllung dieser Bedingung bieten sich folgende Möglichkeiten an:
① Die gesamte ISR befindet sich im Adreßbereich oberhalb des Grund-RAM, so also in einem RAM- oder ROM-Erweiterungsmodul.
② Die ISR ist im Grund-RAM abgelegt, wobei anstelle des abschließenden Befehls RETI ein Sprungsbefehl auf eine Adresse im Bereich zwischen 4000H und 0EFFFH stehen muß. Ab dieser Adresse muß dann die Kodierung für RETI im Speicher abgelegt sein.
Hinweis: Soll die Interruptfreigabe des Prozessors erst nach beendeter ISR erfolgen, so muß der Befehl EI (Interruptfreigabe) unmittelbar vor dem RETI-Befehl stehen. In diesem Fall muß die Befehlsfolge EI, RETI im Bereich zwischen 4000H und 0EFFFH stehen.
Damit für die bei 2. benötigten 2 oder 3 Speicherplätze nicht extra ein RAM-Modul notwendig wird, können die freien Speicherplätze im Bildwiederholspeicher (0EFF1H ... 0EFFFH) dafür genutzt werden.

G. Lützner

Hilfsroutine zur grafischen Bildschirmarbeit

Die gezeigte Routine ist für den KC 85/2 und 3 geschrieben. Sie gestattet, beim ersten Aufruf mittels der Farbbyte den Hintergrund jedes zweiten Zeichens in vertikaler und horizontaler Richtung halb einzufärben. Dadurch ist leicht eine exakte Positionierung auf dem Bildschirm möglich. Wird die Routine erneut aufgerufen, so verschwindet diese Kennzeichnung wieder. Das alles geht ohne Informationsverlust, denn es wird mit XOR 4 gearbeitet.
Die Routine kann per MENÜ mit P aufgerufen werden. Eingebaut in Maschinenprogrammen empfiehlt es sich, die Startadresse 0AH zu verwenden. Der Aufruf ab 04H ist für BASIC gemäß CALL 4 nutzbar. Hierbei wird auf alle Fälle zunächst der Bildwiederholspeicher eingeschaltet.
Die Routine ist voll verschiebbar; kann also auch an jeder anderen Stelle im RAM abgelegt werden. *Prof. Dr. H. Völz*

```
                 ;******************
                 ;     RASTER      *
                 ;H,Voelz  6.3.87  *
                 ; Schaltet auf    *
                 ;Bildschirm Raster*
                 ; ein bzw.. aus   *
                 ;******************
                 ;
'0000  7F7F                 DEFW   7F7FH
'0002  50                   DEFB   'P'
'0003  01                   DEFB   1
                 ;IMR an
'0004  DB88                 IN     A,(88H)
'0006  CBD7                 SET    2,A
'0008  D388                 OUT    (88H),A
'000A  2100A8               LD     HL,0A800H
'000D  0610           M0    LD     B,10H
'000F  7E             M1    LD     A,(HL)
'0010  EE04                 XOR    4
'0012  77                   LD     (HL),A
'0013  23                   INC    HL
'0014  23                   INC    HL
'0015  10F8                 DJNZ   M1-$
'0017  114000               LD     DE,40H
'001A  19                   ADD    HL,DE
'001B  E5                   PUSH   HL
'001C  B7                   OR     A
'001D  1100B0               LD     DE,0B000H
'0020  ED52                 SBC    HL,DE
'0022  E1                   POP    HL
'0023  38E8                 JR     C,M0-$
'0025  0640                 LD     B,40H
'0027  2B             M2    DEC    HL
'0028  10FD                 DJNZ   M2-$
'002A  0604           M3    LD     B,4
'002C  7E             M4    LD     A,(HL)
'002D  EE04                 XOR    4
'002F  77                   LD     (HL),A
'0030  23                   INC    HL
'0031  23                   INC    HL
'0032  10F8                 DJNZ   M4-$
'0034  111000               LD     DE,10H
'0037  19                   ADD    HL,DE
'0038  E5                   PUSH   HL
'0039  B7                   OR     A
'003A  1100B2               LD     DE,0B200H
'003D  ED52                 SBC    HL,DE
'003F  E1                   POP    HL
'0040  38E8                 JR     C,M3-$
'0042  C9                   RET
ERRORS=0000
```

Maschinenprogramme

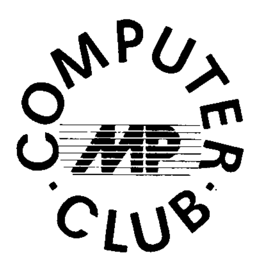
Auch für den BASIC-Programmierer gibt es Fälle, wo es günstig ist, vom BASIC-Programm aus auf Maschinenprogramme zuzugreifen. Hauptanwendungen werden Programmteile sein, für die der Interpreter zu langsam ist. Außerdem können neben Programmen auch Daten abgespeichert werden.
Der BASIC-Interpreter bietet die Möglichkeit, beim Start den Speicherbereich zu begrenzen und somit einen Bereich für Maschinenprogramme freizuhalten. Es gibt aber noch mehr Varianten, die bestimmte Vorteile bieten.
Im Speicher des KC85/3 stehen weiterhin folgende Bereiche zur Verfügung:

0 – 15FH
frei verfügbar
200H – 2FFH
frei verfügbar
BA00H – BFFFH
bedingt verfügbar, dient für Druckertreiber und Zeichengeneratoren.

Der Vorteil besteht darin, daß nichts vom BASIC-Speicher verlorengeht. Der Nachteil bei der Nutzung dieser Bereiche ist, daß die Maschinenprogramme getrennt vom BASIC-Programm abgespeichert und mit der BLOAD-Anweisung geladen werden müssen.
Eine Möglichkeit, das Maschinenprogramm mit dem BASIC-Programm zu verbinden, ist, letzteres in einer Kommentarzeile zu „verstecken". Dabei gibt es aber einiges zu beachten:
– maximale Länge 70 Byte
– das Maschinenprogramm darf kein Byte 00 enthalten
Man geht dabei wie folgt vor:

1. Start des BASIC-Interpreters
2. Eingabe einer Kommentarzeile maximaler Länge mit beliebigen Füllzeichen am Programmanfang
z. B.:
10! ******...***
20! PROGRAMMANFANG
3. Verlassen des BASIC-Interpreters mit BYE
4. Anzeige des BASIC-Programmes
DISPLAY 401 500 4

Durch Betätigen der STOP-Taste kann die MODIFY-Eingabe übergangen werden, und das Maschinenprogramm wird eingegeben. Die Eingabe erfolgt ab Adresse 406H. Die



Adressen 401/402H enthalten den Zeiger auf die Folgezeile, 403/404H sind die Zeilennummer (hexadezimal) und 405H ist die Kennzeichnung als Kommentarzeile. Das Ende einer BASIC-Zeile ist der Kode 00.
Der Aufruf des Unterprogrammes erfolgt mit CALL*406.
Zu beobachten ist, daß beim Listen dieser Programmzeile scheinbar sinnlose Zeichen und Steuerzeichen ausgegeben werden.
Soll ein längeres Maschinenprogramm als 70 Byte eingefügt werden, so muß der Zeiger auf die Folgezeile (401/402H) korrigiert werden.

K.-D. Kirves

SAVE- und LOAD-Routinenaufruf

KC 85/3-BASIC-Tip

Oftmals werden innerhalb eines BASIC-Programmes Daten erzeugt und in einem Speicherbereich außerhalb des BASIC-Programmspeichers abgelegt, die später wieder in einem anderen Programm benötigt werden. Beispiele hierfür sind Zeichengeneratoren oder Funktionstastenbelegungen. Sollen diese abgespeichert werden, so muß der Interpreter verlassen werden, und die SAVE-Routine wird im CAOS aufgerufen. Es gibt aber auch eine Möglichkeit, dies von der BASIC-Ebene aus einem Programm heraus zu tun. Das Bild zeigt in den Zeilen 10 bis 80 ein Programm dafür. In den Zeilen 20 und 30 werden der Anfang und das Ende des Bereiches festgelegt. Im Beispiel sind das:

Anfang: 0 und 185 = B900H
Ende: 156 und 185 = B9BCH

Das ist der Funktionstastenspeicher.

In Zeile 60 enthält eine DATA-Anweisung die Werte für ein kleines Maschinenprogramm, welches in Zeile 70 ab Adresse 0 „gepokt" wird. Die Variable N$ enthält den Namen für die Aufzeichnung, welcher ebenfalls hinter das Maschinenprogramm „gepokt" wird. In Zeile 90 wird dieses Programm aufgerufen.
Soll ein Speicherabzug vom BASIC-Programm geladen werden, kann das mit der BLOAD-Anweisung erfolgen. Dabei ist aber keine Offsetangabe möglich.
Die Zeilen 210 bis 230 enthalten ein Programm dafür.
Hierbei ist kein Maschinenprogramm notwendig. Es müssen nur die entsprechenden Arbeitszellen für das Betriebssystem eingestellt werden. Der Ladeoffset im Beispiel beträgt 100H (auf Adressen AG + 2 = low und AG + 3 = high). Abschließend wird der Programmverteiler auf Adresse F01EH aufgerufen. Näheres dazu findet man in /1/.

K.-D. Kirves

Literatur

/1/ -;Kleincomputer KC85/3 System-Handbuch, Übersichten; VEB Mikroelektronik „W. Pieck" Mühlhausen

```
10 !PROGRAMM ZUM AUFRUF DER SAVE-ROUTINE
20 AG=14208
30 VPOKEAG,54:!UP-NR.
40 VPOKEAG+1,2:VPOKEAG+2,0:VPOKEAG+3,185:!PARAMETERANZAHL,ANFANG
50 VPOKEAG+4,156:VPOKEAG+5,185:!ENDE
60 DATA33,7,0,205,30,240,201:NA$="FTASTEN HEX":!PROGRAMM,NAME
70 FORA=0TO6:READN:POKEA,N:NEXT:!UMLADEN PROGRAMM
80 FORA=1TO8:POKE6+A,ASC(MID$(NA$,A)):NEXT:!UMLADEN NAME
90 CALL0:!AUFRUF SAVE-ROUTINE
200 !PROGRAMM ZUM LADEN VON MASCHINENPROGRAMMEN MIT OFFSET
210 AG=14208:!ADRESSE ARGC
220 VPOKEAG,16:VPOKEAG+1,1:!EINTRAG UP-NR.,PARAMETERANZAHL
230 VPOKEAG+2,0:VPOKEAG+3,1:CALL*F01E:!EINTRAG OFFSET,PROGRAMMVERTEILER
```

DATA-Zeilen-programmgenerator

KC 85/3-BASIC-Tip

Bei der Arbeit mit vielen BASIC-Programmen werden Komponenten benötigt, die als Maschinenprogramme vorliegen und nach dem BASIC-Programm mit der BLOAD-Anweisung geladen werden. Beispiele dafür sind Druckertreiber und Zeichengeneratoren. Besonders in Fällen, wo Programme für Nichtprogrammierer erstellt werden, ist das umständlich und bietet zusätzliche Fehlerquellen. Steht ausreichend Speicher zur Verfügung, was durch die Bereitstellung von 16- bzw. 64-KByte-RAM-Modulen der Fall ist, können die Daten in DATA-Zeilen bereitgestellt und vom BASIC-Programm wieder in den entsprechenden Bereich „gepokt" werden. Die Erarbeitung der DATA-Zeilen ist jedoch recht aufwendig. Aber auch diese Arbeit kann uns der Computer abnehmen.

```
10 !PROGRAMM ZUM ERZEUGEN VON DATAZEILEN
20 INPUT"BEREICH VON A BIS E";A,E
30 INPUT"STARTE DEN KASSETTENRECORDER !";A$
40 OPEN0#1"DATA"
50 B=A:Z=200
60 PRINT#1Z;"DATA";
70 PRINT#1,VPEEK(A);:A=A+1
80 IFA>ETHEN120
90 IFA-B=10*INT((A-B)/10)THEN110
100 PRINT#1",";:GOTO70
110 PRINT#1:Z=Z+2:GOTO60
120 PRINT#1:CLOSE0#1
```

Im Bild ist ein BASIC-Programm dargestellt, welches ein BASIC-Quellprogramm erzeugt, das die benötigten DATA-Zeilen enthält. Da das Datenformat und der -typ bei der PRINT#1- und der LIST#1-Ausgabe prinzipiell gleich sind, kann die Programmerzeugung mit PRINT#1-Anweisungen erfolgen. Der Programmname (Zeile 40) lautet im Beispiel „DATA". In Zeile 20 wird der auszugebende Bereich eingegeben. Die Variable für die Zeilennummern ist Z. Die erste Nummer wird in Zeile 50 festgelegt, der Zeilenabstand in Zeile 110 im Beispiel mit 2. Das Beispiel ist für Daten aus dem Bildwiederholspeicherbereich gedacht, deshalb in Zeile 70 „VPEEK". Soll der Grundspeicherbereich verwendet werden, muß die PEEK-Anweisung benutzt werden. Das so erzeugte Quellprogramm enthält DATA-Zeilen mit je 10 Werten.
Die Einbindung in das Zielprogramm erfolgt mit LOAD#1 „DATA". Dabei werden die Zeilen in ein im Speicher stehendes Programm eingeordnet und können auch bereits vorhandene Zeilen überschreiben. Deshalb sollte die Zeilennumerierung sorgfältig erfolgen. Anschließend müssen im Zielprogramm die Daten wieder auf die ursprünglichen Adressen „gepokt" werden. Gegebenenfalls muß auch eine Initialisierung aufgerufen werden (Druckertreiber), oder bestimmte Arbeitszellen müssen eingestellt werden (Zeichengeneratoren).

K.-D. Kirves

Neue Druckertreiber-programme für KC85-Modul M003 V24

Ergänzend zu den in /1/ beschriebenen Treiberprogrammen wird vom VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen eine erweiterte Kassette C0171/1 V24-Software zum Modul M 003 angeboten. Neben den bereits beschriebenen Programmen enthält die neue Kassette Druckertreiber für die Nadeldrucker K6313 und K6314 und für den Thermodrucker K6304. Die neuen Treiberroutinen arbeiten mit einer Übertragungsrate von 9600 Baud, um die höhere Druckgeschwindigkeit dieser Drucker durch Druckwegoptimierung, besonders im Grafikmodus, voll auszunutzen. Die Druckertreiber arbeiten analog zu denen für die Drucker K6311 und K6312.
Der importierte Matrixdrucker LX86 der Firma EPSON kann ebenfalls mit dem Treiber des K6313, einschließlich der Grafikausgabe, betrieben werden, wobei allerdings Unterschiede bezüglich des Steckverbinders bestehen.
Die Kassette wird zusammen mit dem Modul M003 ausgeliefert. Anwender, die bereits die Kassette C0171 besitzen, können die Kassette C0171/1 als Ersatzteil beim Hersteller bestellen.

K. Schiwon, K.-D. Kirves

Literatur

/1/ Kirves, K.-D.: Serielle Schnittstelle für KC 85/2 und KC 85/3. Mikroprozessortechnik 1 (1987) 4, S. 124
/2/ Schiwon, K.: Ausgabe von Pseudografikzeichen auf Matrixdrucker. Mikroprozessortechnik 1 (1987) 6, S. 180

Nutzerkatalog für Kleincomputer

In Heft 2 der Mikroprozessortechnik wurde der Aufbau eines Nutzerkataloges für die Kleincomputer KC 85/1 und KC 87 angekündigt und um das Angebot nachnutzbarer Hard- und Softwarelösungen zur Aufnahme in den Katalog gebeten. Den sehr zahlreichen Anfragen nach dem Katalog steht zur Zeit leider eine vergleichsweise geringe Anzahl von Lösungen gegenüber, so daß sich die Fertigstellung der ersten Ausgabe verzögert – wir bitten dafür um Verständnis. Im Interesse aller Anwender rufen wir daher nochmals zum Angebot von nachnutzbaren Lösungen an folgende Anschrift auf: VEB Robotron-Meßelektronik „Otto Schön" Dresden, Abt. 1 EKG, PSF 211, Lingnerallee 3, Dresden, 8010.

Dr. G. Kleinmichel

Elektronische Bauelemente

von K.-H. Rumpf, VEB Verlag Technik Berlin 1986, 235 S., 19.50 M

Dieses Lexikon gebräuchlicher elektronischer Bauelemente stellt eine erhebliche Bereicherung der Reihe *1000 Begriffe für den Praktiker* des Verlages dar. Mit großer Sorgfalt wurden 1000 Fachtermini aus der Bauelementephysik, der Bauelementetechnologie und -applikation ausgewählt. Der Text zu jedem der Stichwörter beginnt stets mit einer kursiv gedruckten Definition, die komprimiert in einem Satz wesentliche Aussagen zum Stichwort zusammenfaßt. Daran schließen sich weiterführende Erläuterungen an, die i. allg. sehr detailliert, dabei jedoch nicht weitschweifig sind. In der Regel sind die Ausführungen durch die Angabe der mathematischen Beschreibung von typischen Kennlinienverläufen, Grenzwerten, den Größenordnungen von charakteristischen Parametern der Bauelemente, Nomogrammen, Schaltzeichen, Grundschaltungen, Bauformen und dergleichen mehr ergänzt, so daß auch der Aufbau und die Funktion weniger bekannter, hier besprochener Bauelemente, wie z. B. dem Klystron, dem Peltier-Element oder dem Thyratron, trotz der erforderlichen Kürze der Form anschaulich werden.
Am Schluß des Erläuterungsteils sind gegebenenfalls Verweise auf im Anhang angeführte ausgewählte Normen (DIN, IEC) und Standards (TGL) vorhanden. Diese etwa 250 mit Nummer und Titel zitierten Bestimmungen sind aufsteigend nach ihren Nummern geordnet und zudem vom Autor durchnumeriert worden, um die Verweise in den Texten zu den Stichwörtern zu vereinfachen. Die Sammlung von Bestimmungen gestattet, ohne Anspruch auf Vollständigkeit zu erheben, das leichte Auffinden weiterer Informationen. Zu beachten ist, daß alle hier zitierten Quellen zum Zeitpunkt der Herausgabe des Buches gültig waren, im Anwendungsfall jedoch die Gültigkeit überprüft werden sollte.
Die Handhabung des Buches wird durch fettgedruckte Kopf- und Fußzeilen, die auf das jeweils erste bzw. letzte Stichwort jeder Seite hinweisen, sowie zahlreiche Verweise auf Stichwörter mit zusätzlicher Information erfreulich unterstützt. Es ist sowohl Studierenden von Hoch- und Fachschulen als Nachschlagewerk und auch dem Praktiker bei der täglichen Arbeit und zur Auffrischung vorhandenen Wissens als vom Autor sehr fachkundig und gewissenhaft erarbeitetes Kompendium empfehlenswert.

G. Paulin

Software – Was ist das?

von E. Prager und E. Richter, Verlag Die Wirtschaft Berlin 1986, 80 S., 5,80 M

Um es vorweg zu nehmen: Die Broschüre leistet durchaus einen allgemein verständlichen Beitrag zum Komplex Software in ihren Funktionen als Arbeitsmittel, Arbeitsgegenstand und Produkt. Der Vorzug dieser Schrift besteht darin, allen Interessierten einen politökonomischen Exkurs zur wirtschaftlichen Bedeutung von Software zu vermitteln und zu verdeutlichen, welcher Stellenwert einem Software-Produkt letztlich für die flexible Automatisierung beigemessen werden muß. Die Autoren leiten ihr Anliegen mit Thesen zur wachsenden Rolle der Software ein, die in ihrer Diktion eingängig und überzeugend sind. Die Dialektik von Hardware und Software, und in diesem Kontext die Software als effektivitätsbestimmende Komponente ausdrücklich dargestellt, unterstreicht die Notwendigkeit, Softwareökonomie in größerem Maße als bisher wirksam werden zu lassen. Daß damit dem Software-Herstellungsprozeß, der Multivalenz von Software und Formen ihrer Nachnutzung sowie Grundfragen der Softwarequalität hinreichend Aussagen gewidmet wurden, wird als eindeutige Zwangsläufigkeit für erforderliches Verständnis beim Leser bewertet. Daß Qualifizierung von Kadern bei uns groß geschrieben wird, gilt auch insbesondere für diejenigen, die direkt oder indirekt mit der Produktion, Implementierung, Anwendung, Wartung und Pflege von Software einschließlich der dazugehörigen Leitungsprozesse befaßt sind. Dankenswerterweise haben sich die Autoren auch dieses Problemfeld mit zu eigen gemacht und begründet fixiert: Kader entscheiden alles – auch bei Software. Für jedermann (vielleicht mit Ausnahme der „Profi-Informatiker") eine lesenswerte Einführung in wesentliche Aspekte der Software. Allerdings erscheinen insbesondere zur Bestimmung der Wertgrößen von Software weitere Forschungsarbeiten erforderlich.

Prof. Dr. W. Schoppan

Wörterbuch NO. E.0.3.
Elektronik und Wellenleiter

Englisch – Deutsch / Deutsch – Englisch. Schnellmann Verlag Widnau, Schweiz, 117 S.

Das Wörterbuch gehört zu einer umfangreichen Serie, in der für 6 Sprachen und 8 verschiedene Fachgebiete zweisprachige Kombinationen angeboten werden. Die Aufmachung ist extrem einfach. Auf den einzelnen Seiten fehlt im Kopf die ansonsten übliche Angabe der ersten und letzten Begriffe. Des weiteren wurde infolge des unaufwendigen Druckverfahrens auf jedweden Fettdruck zur Erhöhung der Übersichtlichkeit verzichtet. Grammatikalische Hinweise sind nur in geringem Maße enthalten. Die Erwartungen, die die Angabe des Fachgebietes Elektronik hervorrufen, werden nicht in vollem Umfang erfüllt. Viele in diesem Wissensgebiet bedeutsame Begriffe wie z. B. Schaltkreis, Leiterplatte und Mikroprozessor bleiben unbehandelt. Als Gegenstand neben dem Wellenleiter erscheinen vielmehr die Grundlagen der Elektronik, speziell aus physikalischer Sicht. Und hier ist das referierte Wörterbuch in der Tat erstaunlich inhaltsreich. Der Vergleich mit anderen diesbezüglichen Werken zeigt, daß eine ganze Reihe schwer auffindbarer Spezialausdrücke aufgenommen worden ist. In dieser Hinsicht stellt das Fachwörterbuch des Schnellmann-Verlages eine begrüßenswerte Bereicherung dar.

Prof. Dr. J. Zaremba

Wörterbuch CD.03
Computer und Datenverarbeitung

Englisch – Deutsch / Deutsch – Englisch. Schnellmann Verlag Widnau, Schweiz, 115 S.

Der vorliegende Band ist ein Teil einer zur Zeit 120 Bände umfassenden zweisprachigen Wörterbuchreihe. Die Kombinationen beziehen sich auf die Sprachen Deutsch, Italienisch, Französisch, Englisch, Spanisch und Niederländisch, wobei zur Zeit acht Branchen berücksichtigt werden.

Die Erarbeitung dieser Bände erfolgt rechnergestützt, um mit der schnellen Entwicklung in den einzelnen Fachgebieten und damit auch ihres Wortschatzes Schritt zu halten.

Die inhaltliche Abgrenzung eines Fachwörterbuches ist generell schwierig und kann sicher nie allen Nutzerwünschen entsprechen. Deshalb werden vielfach, so auch im vorliegenden Fall, die Grenzgebiete und vor allem auch die Anwendungsbereiche mit berücksichtigt.

Die im vorliegenden Band zusammengefaßten rund 2500 Begriffe enthalten deshalb auch Vokabeln aus der Elektrotechnik, Regelungstechnik, Analogrechentechnik, Mathematik, Physik u. a. m. Trotz rechnergestützter Bearbeitung sind in dieser 1. Auflage noch eine Reihe von Unzulänglichkeiten erkennbar, die zum Teil störend wirken. So z. B. erfolgt keine Fortsetzung auf einer neuen Zeile, wenn der verfügbare Platz nicht ausreicht; in einer Zeile der linken Spalte stehen zwei verschiedene Vokabeln, ohne daß die zweite in die alphabetische Liste aufgenommen wurde, und schließlich ist auf eine beachtliche Anzahl von Schreibfehlern hinzuweisen.

Außerdem vermißt der Nutzer entsprechende Hinweise, welche ihm die Arbeit mit diesem Fachwörterbuch erleichtern.

Im Hinblick auf zukünftige Auflagen wäre in erster Linie eine Vervollständigung des Wortschatzes des ausgewählten Gebietes anzustreben, damit die gewünschte Zielstellung „eine Ergänzung zu Wörterbüchern der Umgangssprache" noch besser erreicht wird.

Prof. Dr. F. Stuchlik

7. Konferenz der sozialistischen Länder „Magnetische Signalspeicherung"

Die Veranstaltung fand vom 3. bis 8. Mai 1987 in Neubrandenburg statt. Sie wurde traditionell vom ZKI der AdW und der Magnetbandfabrik Dessau (MBF) organisiert. Es nahmen 181 Wissenschaftler und Techniker aus den Akademien, Hochschulen und der Industrie teil. 51 davon kamen aus der UdSSR, ČSSR, VRB, UVR und VRP. In über 80 Vorträgen und einer Postersektion sowie einer Rundtischdiskussion wurde zu den Gebieten der Gerätetechnik (Audio-, Video- und Rechentechnik), zu den Eigenschaften und Technologien von magnetischen Informationsträgern und Magnetköpfen sowie zu den angrenzenden Gebieten – der optischen Speicherverfahren, der Compact-Disc (CD), der Anwendung der Videotechnik beim Film und der optischen Plattenspeicher – beraten. Eine Ausstellung von Sternradio Berlin, Magnetkopfwerk Hartmannsdorf und der Magnetbandfabrik Dessau ergänzten die Beratung.
Am Ende der Konferenz konnte festgestellt werden: Das wissenschaftliche und technische Niveau der Vorträge lag erheblich höher als bei den vorangegangenen Konferenzen. Entsprechend den internationalen Entwicklungen muß die Folge der Konferenzen verdichtet werden; die DDR übernimmt weiterhin den Schwerpunkt und veranstaltet künftig alle 4 Jahre eine Konferenz, dazwischen wird je ein anderes Land die Tagung organisieren. Dadurch geht der bisherige 3-Jahres-Rhythmus in einen 2-Jahres-Rhythmus über. 1989 führt die ČSSR die nächste Tagung durch.
Die gehaltenen Beiträge lassen sich wie folgt gliedern:

Wissenschaftliche Detailprobleme

Hierbei sind insbesondere eine theoretische Vertiefung und umfangreichere Anwendung der Rechentechnik zu nennen. Dabei sind u. a. Probleme des Magnetismus, der Partikelwechselwirkung, Optimierung von Köpfen, Gleiterproblematik, Senkrechtspeicherung, Signalaufbereitung zu nennen.

Übersichtsbeiträge

Sie betrafen vor allem Magnetband, Integrierte Magnetköpfe, Videospeicherung, Audiospeicherung, Audio-PCM, Plattenspeicher, 2-Zoll-Diskette.

Randgebiete,

wie Compact-Disc, optische Speicher, optische Plattenspeicher, Einsatz der Videotechnik beim Film und Halbleiterspeicher. Hierdurch wurde bewirkt, daß die Teilnehmer eine bessere Einordnung ihrer Arbeit vornehmen können.

Bei den digitalen Disketten- und Plattenspeichern wurden u. a. mögliche Anwendungen der 2-Zoll-Diskette, Oberflächenmessungen an Platten, die Streamerproblematik und neue Testprinzipien vorgestellt.
Bei den Informationsträgern wurde über neuartige Magnetbänder auch mit neuen Technologien, von speziellen Untersuchungen zu ihren Eigenschaften bez. Aufzeichnung und Wiedergabe und mechanischen Eigenschaften berichtet.
B. Salzmann analysierte den internationalen Stand und die Weiterentwicklung zu den Plattenspeichern. Dort ging er auch auf die neuesten Entwicklungen und Erfolge bei Robotron ein.
K. D. Hoffmann und *H. Siegel* bemühten sich um eine Analyse neuer Entwicklungen bei den Informationsträgern auf Partikelband. Auch hier wurde der eigene Stand eingeordnet.
Die Langfassungen der meisten Vorträge sind vom ZKI für den Druck vorgesehen.

Prof. Dr. H. Völz

Termine

11. Internationales Seminar über Datenbankbetriebssysteme

WER ? Zentralverwaltung für Statistik der UVR (KSH) und Betrieb für Computer-Anwendungen und -Service
WO ? Seregelyes, Ungarische Volksrepublik
WANN ? 3. bis 10. Oktober 1988
WAS ?
- Methodologien und Hilfsmittel zum Datenbank-Entwurf
- verteilte Datenbanken
- Datenbankmaschinen
- Datensicherheit
- Datenmodelle
- wissensbasierte Systeme
- Datenbanken in CAD/CAM
- Nicht-Standard-Datenbanken

(Veranstaltungssprache: Englisch)
WIE ? Nähere Informationen über: VEB LfA Berlin, Jaques-Duclos-Str. 50/52, Berlin, 1156

Dr. Oßwald



1

59. Internationale Messe Poznan

Traditionsgemäß begann die größte und bedeutendste Ausstellung des polnischen Außenhandels auch in diesem Jahr am zweiten Juni-Sonntag. Sie stand unter dem Leitthema „Material- und energiesparende Techniken und Technologien" und konnte wiederum eine Steigerung der Ausstellerzahl – auf 3850 – verzeichnen.

Tendenzen

Die Computerbranche nahm, wie seit vielen Jahren, auch diesmal breiten Raum ein. Neben dem umfangreichen polnischen Angebot waren in den Länderausstellungen wieder zum Teil namhafte Betriebe und Firmen vertreten. Dennoch schien die im April in Wroclaw erstmals veranstaltete internationale Computerausstellung infosystem '87 Auswirkungen auf die Präsenz vor allem polnischer Aussteller zu haben. Beispielsweise hatte Elwro als bedeutender Hersteller keine Computertechnik ausgestellt. Ähnliches gilt offensichtlich auch für das Offerieren von Neuheiten; vieles hatte man bereits zur infosystem '87 sehen können. Günstig ist eine spezielle Computermesse sicher für die steigende Zahl der meist sehr kleinen Softwarefirmen, die auf der IMP kaum zu entdecken sind. Diese Entwicklung ist in Verbindung mit einer weiteren auf der Messe erkennbaren Tendenz zu sehen: der fast explosionsartigen Zunahme unterschiedlicher Anbieter, z. B. Genossenschaften, Gesellschaften mit beschränkter Haftung usw. Oft sind dies kleine und kleinste Betriebe, die aus importierten Baugruppen – wohl seltener Bauelementen – die Hardware produzieren und komplett mit weit verbreiteter Software vertreiben. Zum Beispiel boten unter der Schirmherrschaft der polnischen Außenhandelskammer in einem eigenen Pavillon mehr als 20 ausländisch-polnische Unternehmen ihre Systeme und Leistungen an. Aufgrund dieser Entwicklung kann die polnische Volkswirtschaft einerseits mit hochwertiger Hard- und Software ausgestattet werden, andererseits wird die Zukunft zeigen, ob die mit der Typenvielfalt verbundenen Fragen des Service gelöst werden. Die unten genannten Firmen refleks und globo beispielsweise existierten zum Messezeitpunkt erst einen bzw. drei Monate, und einige der neuen Firmen sollen sich bereits wieder aus dem Computergeschäft zurückgezogen haben.

Exponate

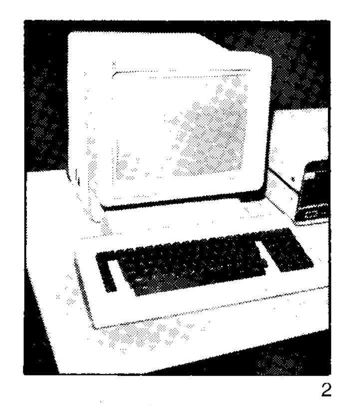
In zwei attraktiven Pavillons (Bild 1) stellte das polnische Außenhandelsunternehmen Metronex GmbH vor allem Erzeugnisse von Betrieben aus, die in der Vereinigung Mera zusammengefaßt sind; z. B. Mera Elzab. Als Neuheit wurde hier das für IBM-PC-kompatible Systeme geeignete Bildschirmterminal **MERA 79152 PC** gezeigt (Bild 2). Einige Daten: Bildschirm 260 × 280 mm^2; 25 Zeilen × 80 Zeichen; Zeichenmatrix 7 × 9 Punkte; Zeichenvorrat 256, kompatibel zum IBM PC XT; Interfaces RS 232 und Centronics; Übertragungsrate 19200 Bit/s, asynchron; die Tastatur umfaßt 98 Zeichen, davon 16 programmierbar (kompatibel zum IBM PC AT).
Als Weiterentwicklung des ComPan 8 war der **ComPan 16** zu sehen (Bild 3). Er hat neben einem 8080-Prozessor zur Arbeit unter CP/M einen 8088 bzw. 8087 (zusätzlicher Arithmetikprozessor) und ist unter PC-DOS zu betreiben. Der RAM beträgt 512 KByte, erweiterbar auf 896 KByte; davon sind 448 KByte als RAM-Disk nutzbar. Es lassen sich bis zu 4 Floppy-Disk-Laufwerke anschließen (5,25 Zoll, 180/360/720 KByte), eine Harddisk mit 20 MByte bzw. – wie auf dem Foto zu sehen – ein Harddisk und

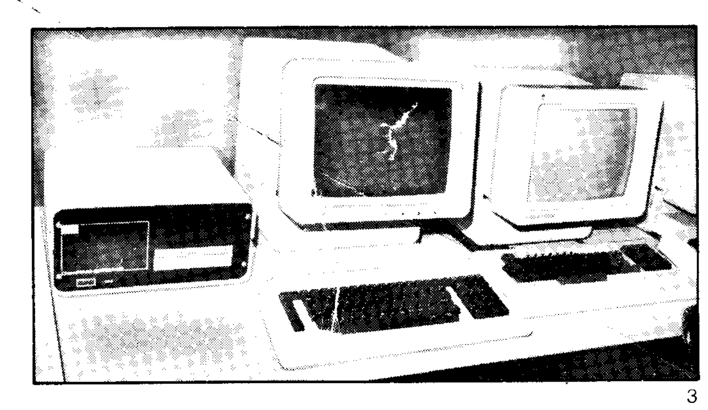
Floppy kombinierendes Beistellgefäß. Als Betriebssysteme kommen MS-DOS bzw. PC/DOS, CP/M, MP/M und Multilink zum Einsatz.
Ebenfalls einen 8-Bit-Vorgänger hat das Echtzeitentwicklungssystem für 16-Bit-Prozessoren **RTDS-16** (Bild 4).
Auf Basis des 8085 zur Steuerung des Systems dient es vor allem der Programmentwicklung von 8086- und 8088-Prozessoren. Der Arbeitsspeicher beträgt 64 KByte RAM, die zusätzliche RAM-Disk umfaßt 256 bzw. 512 KByte; neben der RS-232C-Schnittstelle gibt es vier weitere Interfaces.
Mit einem angeschlossenen Doppel-Diskettenlaufwerk (5,25 Zoll, 100 KByte bei FM, 180 KByte bei MFM) wurde der auf dem Z80 basierende Mikrocomputer **Meritum3** präsentiert (Bild 5). Gegenüber seinen Vorläufern besitzt er z. B. die Möglichkeit der Farbgrafik – 256 × 192 Punkte – und der Vernetzung. Als Betriebssystem kann neben MER-DOS auch CP/M 2.2 genutzt werden.
Eine noch relativ junge Firma ist die Mikrokomputery GmbH, 1985 als Handels- und Produktionsunternehmen auf Initiative von Warschauer Unternehmen gegründet, die an der Entwicklung von Mikrocomputersystemen interessiert sind. Schwerpunkt ist gegenwärtig die Produktion des PC/XT-kompatiblen Mazovia 1016, daneben die Entwicklung verschiedener Peripheriegeräte. Gezeigt wurde als Neuheit die Verbindung mehrerer Mazovia 1016 in einem **TERNET/TRANSNET** genannten lokalen Netz (Bild 6). Die Verbindung erfolgt über RS-232-Schnittstellen mit einer Datenübertragungsrate von 300 bis 19200 Bd. Als Betriebssystem wird DOS 2.0 oder 3.0 verwendet. Die Konfiguration beinhaltete auch einen neuen Farbmonitor mit Grafikfähigkeiten. Ebenfalls neu war der Plotter **GRAF 85** (Bild 7) für A3-Format mit Centronics-Schnittstelle und HP-GL-Möglichkeiten zur Nutzung der Standardsoftware Lotus 1-2-3, Symphonie oder AutoCAD.

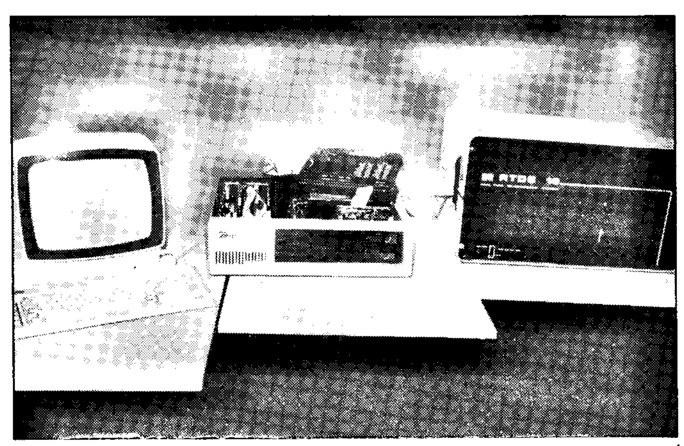
Weitere Daten:
Abmessungen 590 × 490 × 110 mm^3, max. Papiergröße 500 × 350 mm^2, Zeichenfläche 420 × 390 mm^2, Zeichengeschwindigkeit 4 bis 20 cm/s, Schrittweite 0,1 mm, Wiederholgenauigkeit 0,3 mm. Die Masse des Gerätes beträgt 5,8 kg.

Während die Mehrzahl der neugegründeten Firmen PC/XT- und PC/AT- kompatible Systeme im Angebot hatte, ist bemerkenswert, daß einige auch leistungsstarke 32-Bit-Technik offerierten, die zum Teil allerdings nicht als Exponat gezeigt wurde. So die Firma refleks den auf dem NS 32332 basierenden Computer **REFLEKS GENIUSZ** mit 8 MByte RAM, Winchester-Laufwerken bis 376 MByte und Kassetten-Streamern von 60 MByte; die gleiche Firma bot den PC/AT-386 TURBO mit 80386-Prozessor (8, 12 oder 16 MHz), 640 bzw. 1024 KByte RAM und Grafikkarte Hercules an.
Das Unternehmen P. Z. Globo offerierte den BIM/F 386 mit 80386- Prozessor (16 bzw. 20 MHz), 16 MByte RAM und Winchesterlaufwerken bis 240 MByte als demnächst lieferbar.

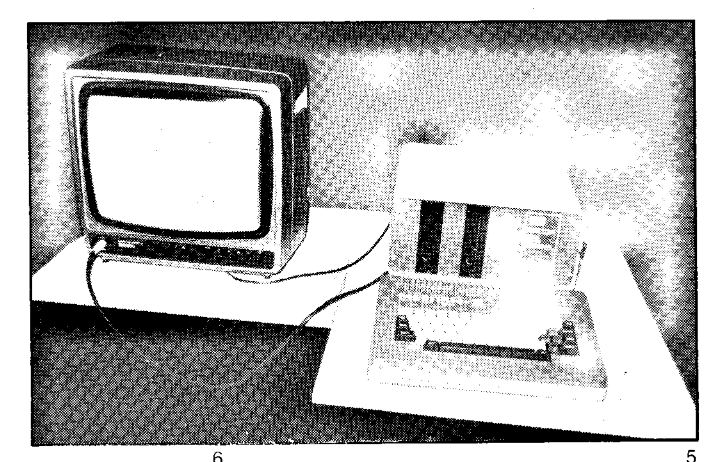
Hans Weiß


2


3



4


6

5



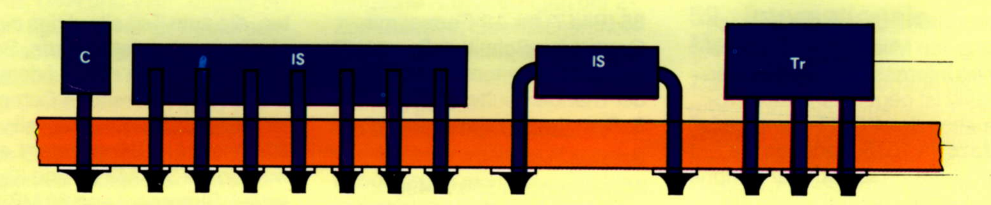

7

Fotos: Weiß (7)

ISSN 0232-2892
Mikroprozessortechnik, Berlin 1 (1987) 10
S. 289–320

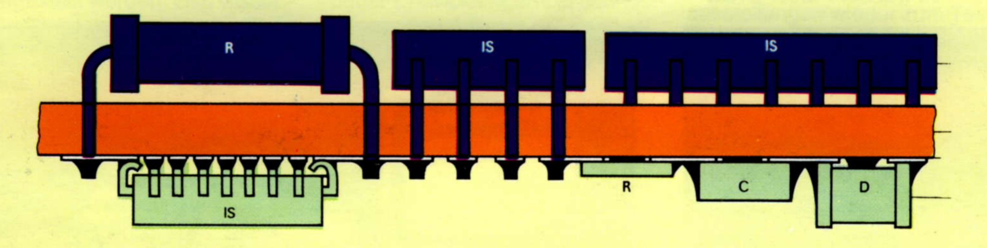
Was ist SMD-Technik?

Bei der bisher gebräuchlichen Leiterplattentechnik werden bedrahtete Bauelemente durch vorbereitete Löcher gesteckt; auf der Rückseite werden die Bauelementeanschlüsse mit den Leiterbahnen verlötet.



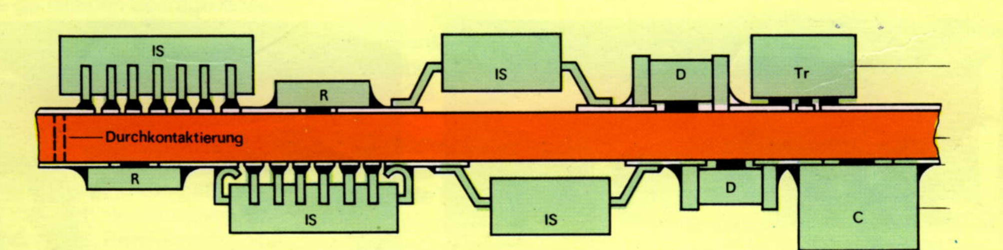
Bestückung einer Leiterplatte mit bedrahteten Bauelementen

Diese traditionelle Leiterplattentechnik wird zunehmend durch die Verarbeitung von SMD ergänzt (SMD – Surface Mounted Devices – oberflächenmontierbare Bauelemente).
Die Bauelemente für die Oberflächenmontage werden auf die Leiterplatte gesetzt, mit geeignetem Kleber mechanisch fixiert und dann im Lötverfahren mit den Leiterbahnen verbunden. In der nächsten Zeit wird die gemischte Bestückung mit bedrahteten Bauelementen für die Oberflächenmontage dominieren.



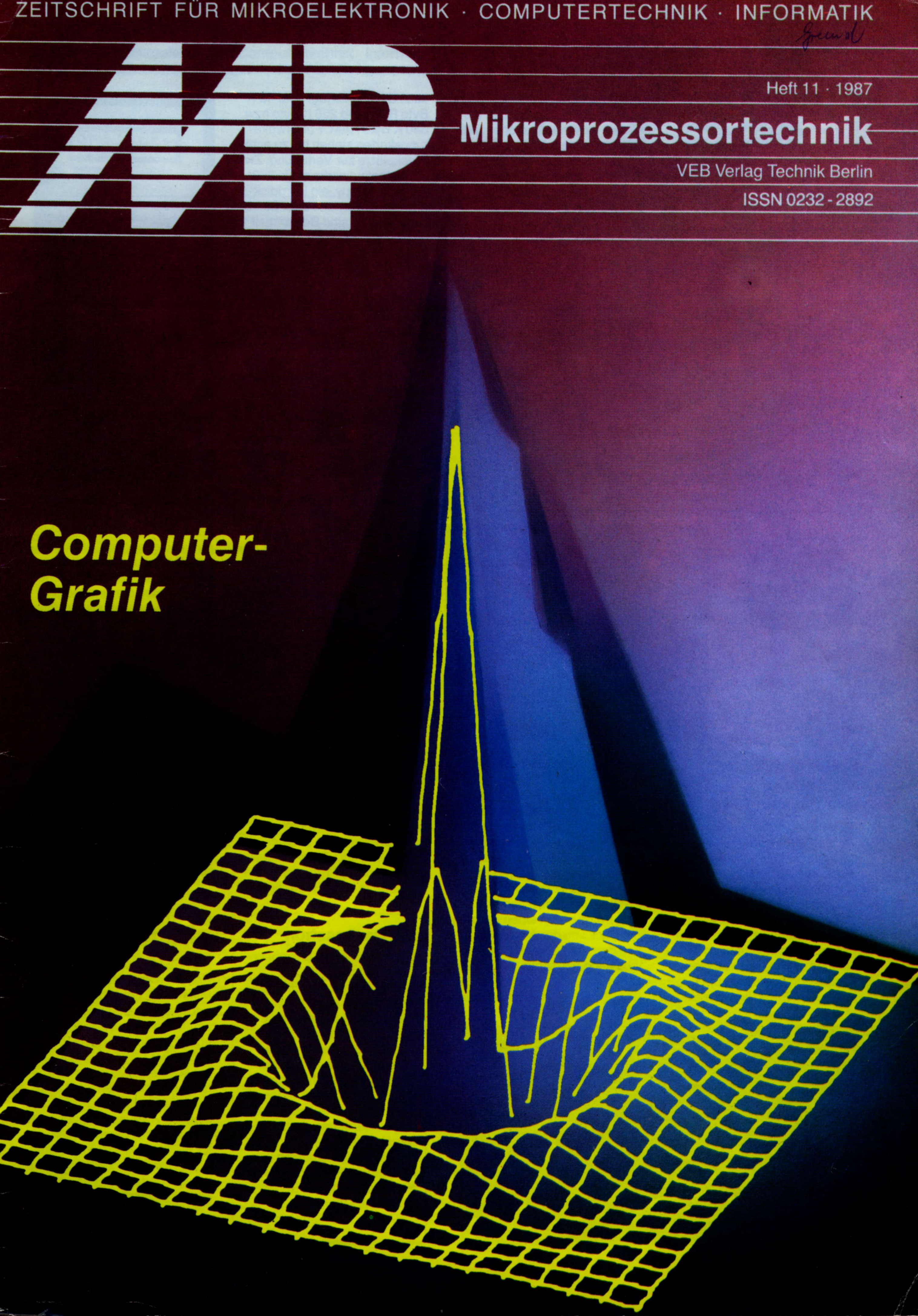
Gemischte Bestückung einer Leiterplatte

Die konsequente Fortführung dieser Technik (SMT – Surface Mounting Technology) ergibt die doppelseitig mit Bauelementen für die Oberflächenmontage bestückte Leiterplatte.



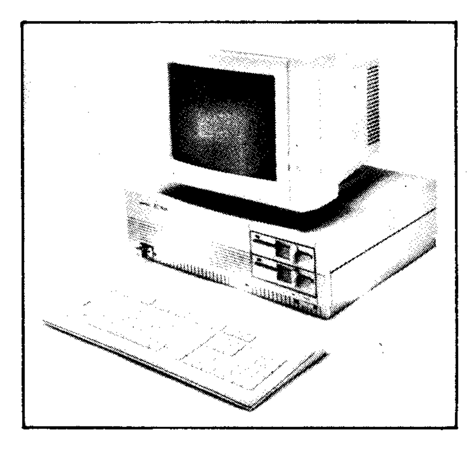
Doppelseitig mit SMD bestückte Leiterplatte

Die Produktivität und Effektivität der SMT ist um so höher, je zielstrebiger und konsequenter die Bauelemente, die Bestückungstechnik und die Fertigungstechnologie aufeinander abgestimmt werden.


ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK
MP
Heft 11 · 1987
Mikroprozessortechnik
VEB Verlag Technik Berlin
ISSN 0232-2892
Computer-
Grafik

ESER-PC

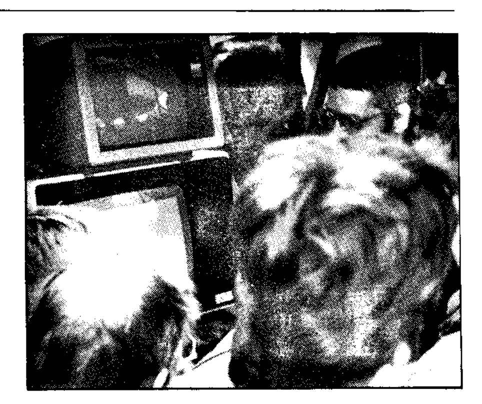
Erstmals auf der Leipziger Herbstmesse 1987 vorgestellt und mit Messegold ausgezeichnet wurde der PC EC 1834 von Robotron. Der PC EC 1834 ist kompatibel zum IBM PC XT und entspricht dem ESER-Standard. Implementiert ist das disketten- bzw. harddiskorientierte Betriebssystem DCP 3.1. Der PC wird in verschiedenen Grundvarianten angeboten. Als Mikroprozessor kommt der K 1810 WM 86 (8086) zum Einsatz. Der PC verfügt über 48 K Byte ROM und 256 K Byte bzw. 640 K Byte RAM. Für Erweiterungskarten stehen bis zu acht Steckplätze zur Verfügung. Zwei Diskettenlaufwerke mit je 360/720 K Byte bzw. ein Harddisk- (bis 40 M Byte) und zwei Diskettenlaufwerke sind Bestandteil der Systemeinheit.
Ein schwenk- und drehbarer monochromatischer, alphanumerischer Bildschirm und/oder ein monochromatischer oder ein Color-Grafikbildschirm können angeschlossen werden. Das System kann mit einem Drucker aus der Reihe robotron K 6313/14, der über Grafik-Modus verfügt, oder einen anderen Drucker mit Centronics-Interface erweitert werden. Weitere Geräte wie Plotter, Digitalisier- und Meßgeräte können über maximal 4 V24- oder IFSS-Schnittstellen angeschlossen werden. Die On-line-Verarbeitung erfolgt nach asynchronem oder synchronem Protokoll (BSCI oder BSCIII) über V.24-Interface. Die Tastatur ist als Flachtastatur für 1 oder 2 Zeichensätze ausgeführt. Als Masse des Gerätes werden etwa 28,5 kg angegeben. Noch in diesem Jahr sollen die ersten 200 Rechner im Robotron Büromaschinenwerk Sömmerda hergestellt werden. *MP*



Werkfoto

Anmerkung der Redaktion: In MP 2/88 beginnen wir unter der Rubrik MP-Kurs mit einer Beitragsfolge zum System K 1810 WM 86 in Einheit von Hardware und Software. Eingegangen wird u. a. auf: Systemarchitektur, Systemschaltkreise, Interfaceschaltkreise, Befehlstypen und Adressierung, Entwicklungshilfen und Assemblerprogrammierung, Programmentwicklung in einer Hochsprache.





Solidaritätsbasar der Berliner Journalisten 1987

Schon zur Tradition geworden ist der große Solidaritätsbasar der Berliner Journalisten, der jedes Jahr am letzten Freitag im August auf dem Alexanderplatz stattfindet. Auch der Verlag Technik war wieder mit einem eigenen Stand vertreten. Viele Betriebe und Institutionen unterstützten die Kollegen unseres Verlages in ihrem Bemühen, einen hohen Erlös für die internationale Solidarität zu erzielen.
Wir bedanken uns insbesondere beim Computerklub des Berliner Hauses der jungen Talente und bei den Kollegen des Konsultationspunktes Mikrorechentechnik der Humboldt-Universität zu Berlin für deren tatkräftige Unterstützung der Redaktion MP. Unsere Fotos sollen einen kleinen Eindruck von der Atmosphäre und dem Andrang, der am Verlagsstand herrschte, vermitteln. *MP*

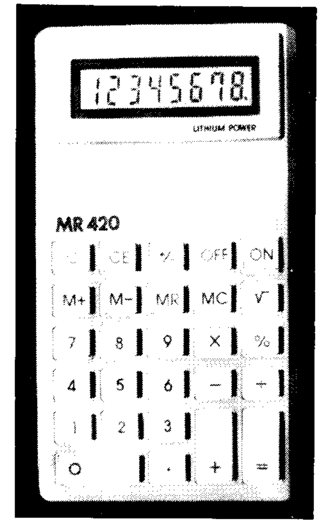
Fotos (2): Paszkowsky



Ein Dankeschön

an alle Leser, die uns auf unseren Aufruf in MP 8/87, S. 228, „Mitarbeit gefragt" geschrieben haben.
Wir hatten Spezialisten gesucht, die die vielen „Tips und Tricks", die der Redaktion von Lesern zugesandt werden, begutachten bzw. testen. So viele schrieben uns, daß wir unmöglich in der Lage sind, allen zu antworten. Wir bitten dafür um Verständnis und bedanken uns nochmals herzlich bei den Einsendern für ihre Bereitschaft zur Mitarbeit.

Ihre Redaktion MP



Werkfoto

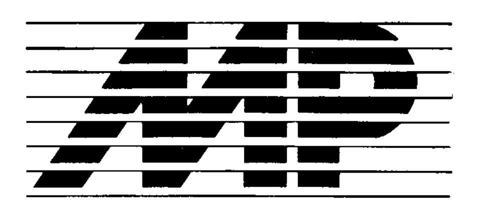
Taschenrechner mit Lithiumbatterie

Weniger als sechs Millimeter hoch ist ein neuer Taschenrechner aus dem VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen. Das neue Geräte wird erstmals mit einer Lithiumbatterie ausgerüstet, die eine Lebensdauer von mehr als sieben Jahren hat. Die hohe Batterielebensdauer ergibt sich aus der stromsparenden 8stelligen LED-Anzeige in Verbindung mit der automatischen Abschaltung (nach etwa 10 Minuten bei Nichtbenutzung). Weitere Vorhaben zielen auf hohe Zuwachsraten in der Taschenrechner- und Diodenfertigung. Auf rund 180 Prozent wird die Produktion von Schulrechnern – SR 1 – im Vergleich zu 1986 steigen. Zwei Millionen Siliziumdioden, die unter anderem für Computer- und Drucktechnik benötigt werden, sollen den Betrieb zusätzlich verlassen. *ADN*

EC 1057 bestand ESER-Test

Der EC 1057 des Kombinates Robotron hat kürzlich in Karl-Marx-Stadt den Abschlußtest erfolgreich bestanden. Ein Gremium von Computerexperten aus Bulgarien, der ČSSR, aus Polen, der UdSSR und Ungarn schätzte nach gründlicher Prüfung die Anlage ein und bestätigte die Produktionsreife. Damit sind die Voraussetzungen gegeben, daß die Neuentwicklung im VEB Robotron-Elektronik Dresden in die Serienfertigung übergeleitet und noch 1987 erstmals exportiert werden kann. *ADN*

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 11 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 2 87 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)
Titel Tatjana Stephanowitz

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

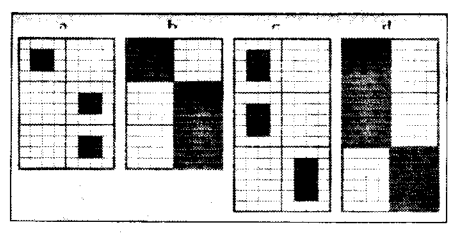
Redaktionsschluß 15. September 1987

AN (EDV) 49837

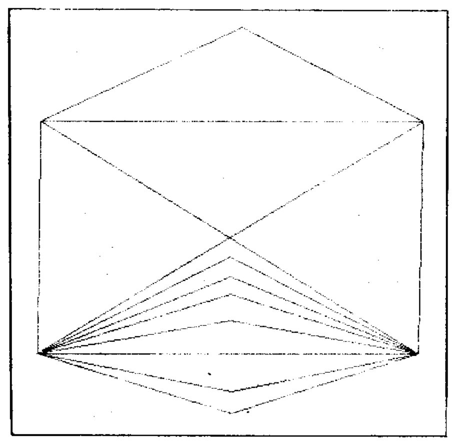
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

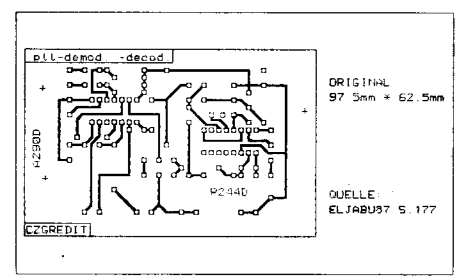
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR–7010, und Leipzig Book Service, Talstraße 29, DDR–7010 Leipzig



Seite 322



Seite 325



Seite 332

Vorschau

In MP 12/87 haben wir für Sie u. a. Beiträge zu folgenden Themen vorgesehen:
- Wirkprinzipien von Informationsaufzeichnungstechnologien
- Plattenspeicher – Stand und Entwicklungstendenzen
- 32-Bit-Mikroprozessoren
- 256-KdRAM-Modul für KC 85/3.

Inhalt

Pseudografik auf dem PC 1715 und dem A 7100

Dr. Hans-Jochen Müller, Dr. Wolf-Dietrich Fromm, Dr. Karl-Heinz Heinig, Dr. Frank Schwarzenberg
Akademie der Wissenschaften der DDR, Zentralinstitut für Kernforschung, Rossendorf

1. Vorbemerkungen

Der PC 1715 hat sich aufgrund der einfachen Handhabbarkeit und der vorhandenen Serviceprogramme ein großes Anwendungsgebiet erschlossen. Auch für wissenschaftliche Berechnungen bietet das breite Spektrum verfügbarer Programmiersprachen Einsatzmöglichkeiten. Der A 7100 eignet sich wegen seines leistungsfähigeren Hardwarekonzepts und der Verfügbarkeit von FORTRAN77 ebenfalls für derartige Anwendungen.

Der wissenschaftliche Einsatz ist vor allem durch einen überwiegenden Anteil an eigenentwickelter Software gekennzeichnet, die Nutzung von Standardsoftwarelösungen steht dabei etwas im Hintergrund. Ein wesentlicher Nachteil für derartige Anwendungen ist das Fehlen von hochauflösenden grafischen Ausgabemöglichkeiten auf dem PC 1715 und der Standardkonfiguration I des A 7100. Grafische Ausgaben auf dem Monitor sind zum einen notwendig für das schnelle Erfassen und Bewerten numerischer Ergebnisse, zum anderen stellt die Computergrafik einen unverzichtbaren Teil der modernen Methode der Computersimulation dar. Bei dieser Methode kommt es nicht nur schlechthin auf die bildliche Darstellung von Ergebnissen an, die Computergrafik stellt selbst eine wesentliche Arbeitsgrundlage der Computersimulation dar, das heißt, sie wirkt interaktiv mit der Anwendersoftware. Die Gestalt der Grafik wird im Programm abgefragt und durch einen von den jeweils wirkenden Naturgesetzen bestimmten Algorithmus weiterentwickelt. Das geschieht schrittweise, so daß die Grafik erst allmählich und im Verlaufe eines oft komplizierten Evolutionsprozesses entsteht. Die Verarbeitung und Wertung der Grafik durch das Programm bildet dafür die Grundlage.

Für solche Anwendungen scheiden die von Kleincomputern her bekannten „Grafiken" mittels vorgegebener Grafiksymbole von vornherein aus, da sie für jedes Einsatzgebiet extra und im allgemeinen mit viel Mühe, das heißt sehr uneffektiv zusammengestellt werden müssen. In /1/ wurde eine Möglichkeit dargestellt, eine Semigrafik auf dem PC 1715 zu installieren. Diese Methode zeichnet sich durch hohe Auflösung, aber auch durch einen entsprechend großen Hardwareaufwand aus, der eine breite Nutzung erschwert /2/. Der freie Steckplatz des PC 1715 geht dafür verloren und steht z. B. für eine RAM-Disk-Erweiterung nicht mehr zur Verfügung /3/. Voraussetzung für die Anwendbarkeit dieses Verfahrens ist, daß sich die Grafiken in höchstens 128 verschiedene, dem Zeichenraster entsprechende Grundelemente zerlegen lassen, was im allgemeinen Fall nicht erfüllt ist.

2. Konzept der Pseudografik

Bei dem Konzept einer Pseudografik für den PC 1715 und der alphanumerischen Variante des A 7100 wurde von den folgenden Prinzipien ausgegangen:

- Es soll kein zusätzlicher Hardwareaufwand entstehen. Diese Rechnertypen wurden nicht für eine hochauflösende Grafik vorgesehen. Eine nachträgliche Veränderung wäre mit umfangreichem Hardwareaufwand verbunden.
- Eine Punktgrafik mittlerer Auflösung auf der Basis von pseudografischen Symbolen stellt eine angepaßte Erweiterung dar. Sie ist eine abgerüstete, aber universelle Grafikvariante, die sich für die interaktive Arbeitsweise eignet.
- Wegen der beschränkten Auflösung erscheint eine Perfektionierung (z. B. unterstützte Beschriftung oder maßstäbliche Darstellung) nicht sinnvoll. Der Softwareaufwand soll den vorhandenen Speicherplatz nur unwesentlich beeinflussen.
- Die Grafik soll sowohl bei der Arbeit mit Interpretern als auch mit Compilern verfügbar sein. Dabei kommt es auf weitestgehende Kompatibilität im Aufruf der Pseudografik vom PC 1715 und vom A 7100 aus an, damit keine Programmänderungen bei Programmtransfer notwendig sind.
- Zur Auswertung von Grafiken gehört die Möglichkeit, vom Monitorbild eine Hardcopy anzufertigen, die in speziellen Fällen eine gekoppelte Ausgabe von grafischen und alphanumerischen Symbolen gewährleistet.

Der alphanumerische Bildschirm umfaßt 80 × 24 (80 × 25) Zeichen (Daten des A 7100 im folgenden in Klammern eingeschlossen). Dabei besteht jedes Zeichen aus 8 × 12 (8 × 16) Bildpunkten. Das entspricht einem notwendigen Speicherplatz von 12 (16) Bytes zur Generierung eines Zeichens.

Wollte man jeden einzelnen Bildpunkt im Rahmen einer Pseudografik nutzen, würde das einen Zeichensatz von 2^{96} (2^{128}) unterschiedlichen Zeichen erfordern. Eine Reduzierung der Auflösung ist deshalb notwendig. Neben den 128 alphanumerischen Zeichen kann der Computer 128 verschiedene Grafiksymbole ansteuern. Damit ist im Rahmen einer Pseudografik die Grenze der Auflösung gegeben. Unter Berücksichtigung der Symmetrie der Grafiksymbole und des verfügbaren Zeichenumfanges bietet sich eine Zerlegung eines Zeichens in 2 × 3 Felder an. Die Summe aller Kombinationen von belegten und leeren Feldern entspricht einem Umfang von 64 Zeichen. Die Auflösung der Pseudografik beträgt dann 160 × 72 (160 × 75) „Punkte". Da mit den notwendigen 64 Zeichen nur die Hälfte des ansteuerbaren Vorrates genutzt würde, wurden zwei unterschiedliche Zeichensätze pro Computer implementiert:

- Beim PC 1715 besteht jedes Pseudografikfeld aus 4 × 4 Pixel. Der erste Zeichensatz realisiert zentrierte „kleine Punkte", bestehend aus jeweils 2 × 2 Pixel innerhalb der 4 × 4-Matrix, während beim zweiten Zeichensatz das gesamte quadratische Feld ausgefüllt wird („große Punkte"; Bild 1a und b).
- Beim A 7100 ist die Situation komplizierter. Die 8 × 16 Pixel eines Zeichens wurden in 4 Felder zu je 4 × 5 Pixel und 2 Felder zu je 4 × 6 Pixel unterteilt. Dementsprechend besteht der kleine Punkt aus 2 × 3 Pixel (wobei dieser Punkt in der 4 × 6-Matrix vertikal nicht zentriert ist), während bei großen Punkten die Fläche eines Feldes voll ausgefüllt ist (siehe Bild 1c und d).

Mit Hilfe der beiden Zeichensätze können unterschiedliche Aufgaben gelöst werden. Kleine Punkte eignen sich besonders für die Darstellung gekrümmter Linien, wogegen große Punkte insbesondere für vertikale und horizontale Geraden verwendet werden können (z. B. Koordinatenachsen oder Balkenhistogramme). Beide Zeichensätze sind innerhalb einer Grafik verwendbar. Die sechs Felder eines Zeichens gehören allerdings zu einem Zeichensatz. Eine Mischung großer und kleiner Punkte innerhalb eines Zeichens ist nicht möglich. Wird eine Mischung innerhalb eines Zeichens versucht, so haben kleine Punkte Priorität, das heißt, alle belegten Felder des Zeichens werden als kleine Punkte wiedergegeben.

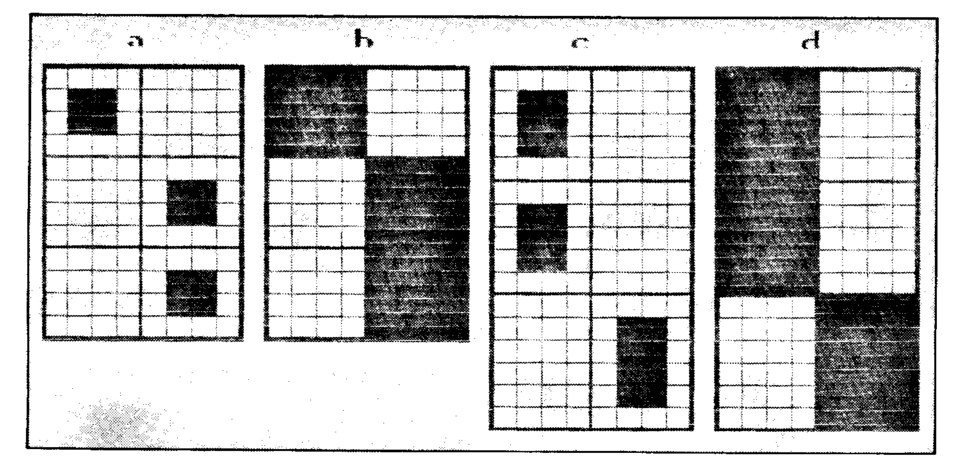


Bild 1 Beispiele für verfügbare Pseudografiksymbole am PC 1715 (a und b) und am A 7100 (c und d)

3. Softwarelösung

3.1. PC 1715

Zur Realisierung der Pseudografik auf dem PC 1715 gehört das Einsetzen eines 2-KByte-EPROM auf den vorhandenen freien Steckplatz der Rechnerplatine unterhalb der Diskettenlaufwerke. Dieser EPROM enthält die Information für die Generierung der Pseudografikzeichen. Das Stecken dieses 2. Zeichengenerators stellt die einzige Veränderung am PC 1715 dar.
Für die Speicherung der Pseudografik wird der Bildwiederholspeicher des PC 1715 genutzt. Das hat den Vorteil, daß die Dynamik der Grafik auf dem Bildschirm verfolgt werden kann (wichtig für Computersimulation!). Der Bildschirmwiederholspeicher belegt standardmäßig 2 KByte RAM ab Adresse F800H, kann aber innerhalb des TPA in Abständen von 800H zu niedrigeren Adressen verlegt werden. Für eine alternierende Ausgabe von alphanumerischen Daten und Grafik ist es günstiger, für die Grafik einen zusätzlichen Bildspeicher von 2 KByte anzulegen und bei der Ausgabe zwischen beiden umzuschalten /4/. Diese Möglichkeit wurde beim Softwarekonzept berücksichtigt. Wahlweise kann die Grafik im Standardbildspeicher (zugehörige Programme sind PLOTF8 bzw. RUBF8) oder ab B000H (Bezeichnung PLOT bzw. RUB) generiert werden. Allerdings muß der Nutzer gewährleisten, daß dieser Bereich nicht durch ein anderes Programm belegt wird.
Die Ansteuerung der Punkte wurde analog zur Organisation einer hochauflösenden Grafik mit Bit-Mapping-Prinzip realisiert. Jedem Feld innerhalb eines Zeichens wurde ein Bit zugeordnet. Der Zeichencode ergibt sich dann aus der entsprechend den gesetzten Bits gebildeten Dualzahl (0...63). Durch diese Zuordnungsvorschrift reduziert sich der Algorithmus zum Setzen, Löschen und Abfragen (als Testen bezeichnet) einzelner Grafikpunkte im wesentlichen auf die Adreßberechnung des Zeichens im Bildspeicher und des aktuellen Feldes im Zeichen. Diese Funktionen wurden innerhalb des U880-Assemblerprogramms PLOT bzw. PLOTF8 realisiert. Das Programm wird als Unterprogramm mit 3 Parametern aufgerufen: X- und Y-Koordinate und ein Steuerparameter (siehe Tafel 1). Bei Bereichsüberschreitung der X- und Y-Koordinate unterdrückt das Programm die Ausführung. Für die Funktionsauswahl sind nur die drei niederwertigen Bits des Steuerparameters relevant. Als Entry RUB bzw. RUBF8 wurde zusätzlich das Löschen des jeweiligen Bildspeichers impliziert.
Das Programm liegt als linkbares REL-File (PLOT.REL) vor und ist für alle Compilersprachen verwendbar. Zur Nutzung unter dem BASIC-Interpreter muß das Programm ab einer Absolutadresse geladen werden. Das wird durch den COM-File PLOTSCP.COM realisiert, der vor dem Laden des BASIC-Interpreters aufgerufen werden muß. Zur Vermeidung von Überschreibungen wurde dabei für die Speicherung des Programms die ungenutzte 25. Zeile des Standardbildwiederholspeichers verwendet (ab Adresse FF81H). Unter dem Betriebssystem CP/A muß das Programm allerdings in einen Adreßbereich im TPA geladen werden. Dafür

Tafel 1 Darstellung der implizierten Software zur Ansteuerung der Pseudografik (A: PC 1715; B: A 7100)

Name	Parameter	Typ	Funktion	Bem.
RUB	–	–	Löschen des Grafikspeichers	A+B
RUBF8	–	–		
SET	–	–	Belegen des Grafikspeichers mit Punkten	B
SETF8	–	–	Belegen des Grafikspeichers mit Rechtecken	B
	IX=0...159	I*2	X-Koordinate	A+B
	IY=0...72	I*2	Y-Koordinate	A
	IY=0...74	I*2	Y-Koordinate	B
PLOT PLOTF8	IT= 0	I*2	Punkt löschen	A+B
	1		Setzen Quadratpunkt	A+B
	2		Punkttest; Rückgabe: IT=0 leer, IT≠0 gesetzt	A+B
	3		Darstellung der Grafik auf Display/Hardcopy	B *)
	4		Punkt löschen	A+B
	5		Setzen Punkt	A+B
	6		Punkttest (wie IT=2)	A+B
	7		Laden Pseudografik	B *)
	11		Hardcopy	B *)
HARDC	–	–	Hardcopy	A

Bemerkungen:
*) Diese Funktion wurde gegenüber dem PC 1715 erweitert. Beim Aufruf am PC 1715 erfolgt keine Reaktion.

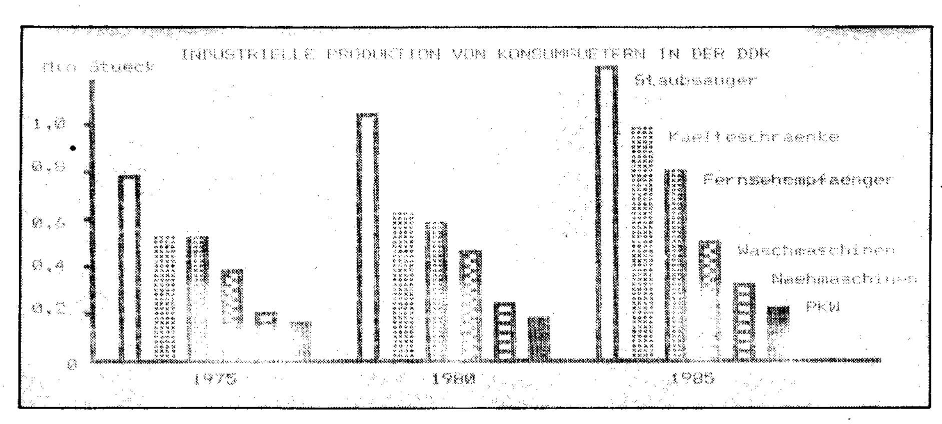
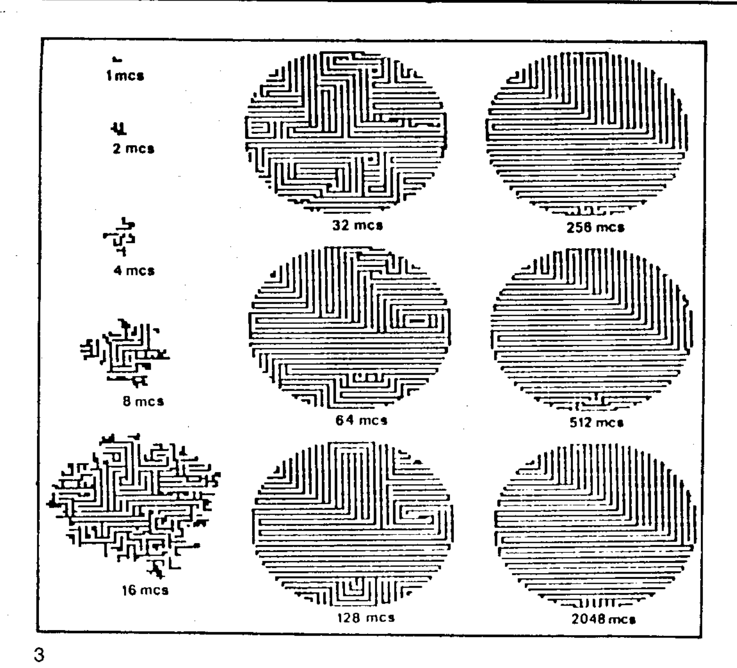


Bild 2 Einsatz der Pseudografik bei der Darstellung von Histogrammen am PC 1715 (Quelle: Statistisches Jahrbuch der DDR 1986)

wird die 25. Zeile des Zusatzbildspeichers verwendet.
Das File PLOTCPA.COM lädt PLOT ab Adresse B781H. Die Verwendung der PLOT-Routinen bei Nutzung von Compilern und unter BASIC ist in den Beschreibungen PLOTCOMP.DOK und PLOTINT.DOK dokumentiert.

3.2. A 7100

Zur Erarbeitung der Software für die Pseudografik am A 7100 war ein anderes Konzept erforderlich. Durch das Datenbus-Prinzip des A 7100 ist ein direkter Zugriff auf den Bildwiederholspeicher in der ABS über die Software nicht möglich. Außerdem können dadurch keine Zeichen aus dem Bildschirmspeicher abgerufen werden. Deshalb ist es notwendig, zwischen die Anwendersoftware und die ABS eine Vermittlungsroutine einzufügen, die alle notwendigen Aufgaben zur Generierung einer Pseudografik übernimmt und diese bei Bedarf auf dem Bildschirm darstellt. Damit ist ein wesentlicher Nachteil der Pseudografik auf dem A 7100 gegenüber dem PC 1715 gegeben: Die dynamische Entwicklung der Pseudografik kann nicht mehr auf dem Bildschirm verfolgt werden. Es ist nur eine intervallweise Darstellung möglich. Zum anderen muß zusätzlich zum Bildwiederholspeicher in der ABS ein Grafikzwischenspeicher im TPA angelegt werden, der eine interaktive Arbeit zwischen Anwendersoftware und Grafik realisiert. Bei der Konzipierung dieses Speichers mußte zwischen Speicherbedarf und notwendiger Rechenzeit optimiert werden. Es ergab sich ein Speicherbedarf von etwa 10 KByte.
Aus Gründen der Softwarekompatibilität zum PC 1715 wurde eine Subroutine erstellt, die alle Funktionen und Entries der Grafikfunktion des PC 1715 erfüllt. Auch die unterschiedlichen Aufrufe PLOT bzw. PLOTF8 wurden realisiert, obwohl ihnen hier keine Bedeutung zukommt. Einige zusätzliche Funktionen wurden jedoch notwendig, die durch die intervallweise Darstellung der Pseudografik und das Laden der Pseudografiksymbole durch Software bedingt sind. Diese Zusätze sind in Tafel 1 gekennzeichnet. Zur Erweiterung des Service wurden die Funktionen SET bzw. SETF8 in die PLOT-Routine aufgenommen.
Zur Verkürzung der Rechenzeit wurde auf eine interne Fehlerbehandlung verzichtet. Bei Grenzwertüberschreitung der X- bzw. Y-Koordinaten erfolgt ein Programmabbruch mit Feldüberlauf-Hinweis. Zusätzlich ist gewährleistet, daß bei unzulässiger Parametereingabe die Grafik nicht zerstört wird.



3

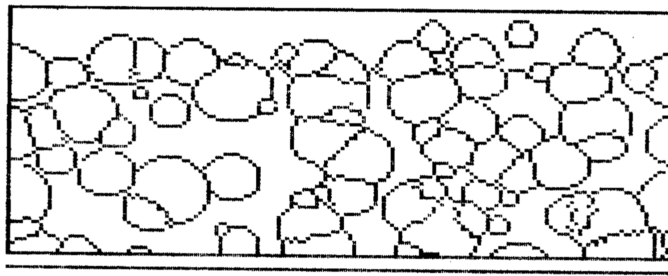

4

Bild 3 Pseudografik bei der Computersimulation der Evolution von Schmelzinstabilitäten (PC 1715; mehrere Bilder zusammengefaßt)
Bild 4 Simulation von Kristallisationsprozessen mit Hilfe der Pseudografik am A 7100 (Die untere Linie gibt die volle Größe der Grafik an.)

Für die Arbeit unter FORTRAN77 benötigt die Subroutine die Funktion BDOS86 und zum Generieren der Pseudografiksymbole die Subroutine PSEUDO, die beim Linken mit angegeben werden müssen. Zur einfachen Handhabung wurden alle benötigten Files in einer Bibliothek mit der Bezeichnung PGRAF.L86 zusammengefaßt. Vor der ersten Darstellung der Pseudografik (Parameter = 3) muß der Pseudozeichensatz generiert werden (Parameter = 7). Er kann aber auch direkt, durch den Aufruf von PSEUDO.CMD, aktiviert werden. Eine Verwendung der Pseudografik im BASIC-Interpreter des A 7100 ist möglich, erfordert aber das Laden der notwendigen Subroutinen als Assemblerprogramme an feste Programmadressen. Darauf soll hier aus Platzgründen nicht näher eingegangen werden. Dies ist in /5/ beschrieben.

4. Hardcopy der Pseudografik

Zur Weiterverwendung der Grafiken wurden für beide Rechner Hardcopyfunktionen erstellt, die eine schnelle und unkomplizierte Ausgabe der Monitor-Pseudografik mit dem Nadeldrucker LX-86 garantieren. Aufgrund der bisherigen Einsatzgebiete wurden unterschiedliche Ausführungen realisiert. Die Hardcopyfunktion des PC 1715 wurde in PASCAL geschrieben und steht als selbständig aufrufbares File unter der Bezeichnung HARDC.COM zur Verfügung. Realisiert wurde die gemischte Ausgabe von grafischen und alphanumerischen Informationen. Die Hardcopyfunktion des A 7100 ist in die PLOT- bzw. PLOTF8-Funktion eingebunden und kann durch entsprechende Parameterwahl abgerufen werden. Zusätzlich besteht die Möglichkeit, während der Bildschirmdarstellung der Pseudografik durch Drücken einer beliebigen Taste in die Hardcopyfunktion zu verzweigen. Dabei kann zwischen einfacher und doppelter Größe bzw. Dichte der Grafik auf dem Nadeldrucker gewählt werden. Ein externer Aufruf der Hardcopyfunktion ist am A 7100 nicht möglich, da die Pseudografik nur innerhalb der Programmabarbeitung existiert. Zur Schonung des Nadeldruckers wurde eine automatische Bewertung der Grafik implementiert, die gewährleistet, daß immer eine Darstellung mit minimaler Anzahl von zu druckenden Punkten gewählt wird (normale oder inverse Ausgabe).

5. Einschätzung

Mit der entwickelten pseudografischen Ausgabemöglichkeit auf dem PC 1715 und der alphanumerischen Variante des A 7100 wurde ein Hilfsmittel geschaffen, das die Einsatzmöglichkeiten der Rechner in Wissenschaft, Forschung und Büroautomatisierung erweitert. Die Grafik wurde von vornherein als Minimalvariante (sowohl vom Hard- als auch vom Softwareaufwand) konzipiert und nicht auf spezielle Anwendungen zugeschnitten. Sie ist aufgebaut wie eine Vollgrafik und daher universell einsetzbar. Weitere Merkmale sind einfache Handhabbarkeit und die Möglichkeit der interaktiven Arbeitsweise zwischen Anwendersoftware und Pseudografik.
Der einzige Eingriff in den Rechner besteht im Stecken eines 2-KByte-EPROM auf den vorhandenen Steckplatz im PC 1715.
Die Einsatzgebiete sind bei beiden Rechnern durch die generell begrenzte Auflösung der Grafik eingeschränkt. Für den PC 1715 wurde die Adressierung und Zeichengenerierung im Bildspeicher als Assemblerroutine geschrieben, so daß die benötigte Zeit für die Bildschirmdarstellung in der Regel vernachlässigbar ist gegenüber der Rechenzeit des Nutzerprogramms. Demgegenüber schränken für den A 7100 die Zeiten für das Setzen eines Punktes (1 ms) und für einen kompletten Bildaufbau (17 s) die Anwendbarkeit der Pseudografik insbesondere auf dem Gebiet der Computergrafik ein. Hier ist noch eine wesentliche Beschleunigung der Grafik durch Programmierung in C oder Assembler (derzeit FORTRAN77) möglich.
Die entwickelte Software (einschließlich Beschreibung) und das Bitmuster für den 2. Zeichengenerator des PC 1715 stehen Interessenten zur Nachnutzung zur Verfügung.

Literatur

/1/ Herden, D.; Lüdicke, R.; Wippich, K.: Semigrafik für PC 1715. Mikroprozessortechnik, Berlin 1 (1987) 1, S. 19
/2/ Leserforum Pseudografik 1715. rechentechnik/datenverarbeitung, Berlin 24 (1987) 5, S. 4
/3/ Ihlenfeld, A.; Riedel, W.: RAM-Floppy RAF 512 für K 1520. Radio, Ferns., Elektron., Berlin 36 (1987) 4, S. 268
/4/ Systemhandbuch SCP. VEB Robotron-Büromaschinenwerk Sömmerda, Stand 31. 12. 1984
/5/ BASIC-Interpreter, AC A 7100. VEB Robotron Projekt Dresden C 1015-0300-1 M 3030

✉ KONTAKT ☎

Akademie der Wissenschaften der DDR, Zentralinstitut für Kernforschung Rossendorf, Postfach 19, Dresden, 8051; Tel.: 5 91 24 45 oder 5 91 23 77

Kyrillischer Druck auf LX-86

Das Programm EPRUSS erlaubt auf dem Drucker LX-86 (z. B. am PC 1715) den wahlweisen Druck kyrillischer bzw. lateinischer Zeichen auf einem Blatt. Das heißt, es kann über eine frei wählbare Spaltenbreite ein russischer Text gedruckt werden, während daneben z. B. die deutsche Übersetzung stehen könnte. Das Schreiben geschieht am besten mit dem Textverarbeitungssystem TP (bzw. WORDSTAR). Voraussetzung ist ein CP/M-kompatibles Betriebssystem.
VEB Neptunwerft Rostock, Abt. TN, Karl-Liebknecht-Straße, Rostock, 2500; Tel. 3 84 30 59 (Wesolek).
Straub/Wecker

Installation für LX-86

Um die Möglichkeiten der Druckbildgestaltung des Epson LX-86 am PC 1715 optimal nutzen zu können, ist eine entsprechende Installation des Druckers notwendig. Die Bedienelemente gestatten nur die Einstellung eines geringen Teiles dieser Druckparameter.
Mit INSTLX wurde ein Softwarewerkzeug erarbeitet, welches die Änderung weiterer Installationsparameter im Bildschirmdialog gestattet und damit eine optimale Nutzung der Drucktechnik ermöglicht. Folgende Einstellungen können realisiert werden:

Seitenlänge · linker Rand · unterer Rand · Zeilenabstand · Kombination von Schriftarten · Wahl spezieller Zeichensätze · Ein-/Ausschalten Papierendefühler · Ein-/Ausschalten Grafikzeichensatz

Somit lassen sich praxisrelevante Druckfunktionen über Hauptmenü schnell und unkompliziert ändern. Die Programmausführung erfolgt unter SCP bzw. CP/M.
Deutsche Post, Studiotechnik Fernsehen, PN, Rudower Chaussee 3, Berlin, 1199; Tel 6 73 73 24 *Feustel*

Grafik am A 7100

Ottmar Vetter
VEB Elektroprojekt und Anlagenbau Berlin

1. Vorbemerkungen

Besonders bei der Anwendung höherer Programmiersprachen besteht verstärkt die Notwendigkeit, die grafischen Fähigkeiten des vorhandenen Arbeitsplatzcomputers zu nutzen.
Der 16-Bit-Arbeitsplatzcomputer A 7100 besitzt eine sehr leistungsfähige Schwarz-Weiß-Graphik. Das Grafik-Subsystem /1/, bestehend aus Grafikkontroller K7070 (KGS) und der Anschlußsteuerung für den Bildschirm K7072 (AGB), ist unter anderem durch die hohe Auflösung von 640×400 Bildpunkten in vier Helligkeitsstufen gekennzeichnet.
Für Nutzer des Betriebssystems SCP 1700 wird die Grafikerweiterung SCP-GX bereitgestellt /2/. Sie besteht aus dem hardwareunabhängigen Teil, dem Graphics Device Operating System (GDOS), und dem hardwareabhängigen Teil, dem Graphics Input/Output System (GIOS). Über das SCP-GX lassen sich grafische Ein- und Ausgaben von und zu verschiedenen Geräten (z. B. grafisches Tablett, Bildschirm) realisieren.
Die vorgestellte Grafikimplementierung entstand in Zusammenarbeit mit der Technischen Universität Dresden (Sektion Elektrotechnik, Bereich Automatisierungstechnik) und findet in der digitalen Simulation von Elektroantrieben Anwendung.
Die Nachnutzung des TURBO-PASCAL Programmes TURBOGX.PAS einschließlich einer ausführlichen Dokumentation zur Arbeitsweise der Proceduren sowie einiger Beispiele ist über die Kontaktadresse möglich.

2. Übergang in den Grafikmodus

Voraussetzung für die Nutzung des SCP-GX ist, daß die ladbare Grafikfirmware in den KGS geladen wurde. Befindet sich die aktuelle Version der Grafikfirmware auf Laufwerk A (z. Z. GRAF5.FRM), erfolgt die Kommandoeingabe.

A > L GRAF6.FRM

Nun ist es möglich, das Betriebssystem von der Kommandoebene aus in den Grafikmodus zu setzen bzw. rückzusetzen (Unterschied zum Grafikmodus ein- bzw. ausschalten). Mit der Kommandoeingabe

A > GRAPHICS ⟨LW⟩:

wird das Betriebssystem in den Grafikmodus gesetzt, wobei sich auf dem mit LW spezifizierten Laufwerk die Gerätetreiber (z. B. für den Bildschirm – KGSTBDRV.SYS) und die Zuweisungstabelle ASSIGN.SYS befinden müssen.
Nach jedem Hardware-Booten muß dieses Kommando wiederholt werden, um in den Grafikmodus zu gelangen. Der durch SCP-GX zusätzlich belegte Speicherraum verringert den Speicherbereich für Nutzerprogramme. Außerdem ist für Nutzerprogramme ein genügend großer Stackbereich vorzusehen /2/. Will man den Grafikmodus verlassen, so geschieht das mit dem Kommando

A > GRAPHICS NO

und der von SCP-GX belegte Speicher wird an das System zurückgegeben.

3. Funktion von SCP-GX

Der Zugriff zu grafischen Funktionen erfolgt über eine herkömmliche BDOS-Ruffolge. Vom Nutzerprogramm wird GDOS über das Softwareinterrupt 224 mit dem Funktionscode 473H im Register CX aufgerufen. Außerdem wird eine Parameterliste /2/ übergeben, wobei deren Adresse mit den Registern DS (Segmet) und DX (Offset) übergeben wird. Die Parameterliste enthält 5 Adressen für spezielle Parameterbereiche, die an das

Bild 1 Grafikprozeduren

GRAFIK_EIN (<Gerät>) <Gerät>	Dieses Kommando schaltet den Grafikmodus für das mit <Gerät> festgelegte Gerät ein. 'Bildschirm' - Bildschirm als Ausgabegerät 'Drucker_HR' - K6313 mit hoher Auflösung 'Drucker_LR' - K6313 mit geringer Auflösung 'Plotter' - Plotter K6411
GRAFIK_AUS	Das Kommando schaltet den Grafikmodus aus. Alle noch anstehenden grafischen Kommandos werden ausgeführt. Alschließend wird in den Alphanumerikmodus übergegangen.
CLR_GRAFIK	Der grafische Bildwiederholspeicher wird gelöscht.
HARDCOPY	Das standardmäßige Hardcopygerät (Drucker) wird zum Kopieren des Bildschirminhaltes veranlaßt.
MARKER (X,Y)	Ein Markersymbol wird an die mit dem Koordinatenpaar X,Y festgelegte Position gezeichnet. Vorher müssen die Markerattribute festgelegt worden sein.
MARKER_HOEHE (M_HOEHE)	Markerattribut - die Markerhöhe wird in Geräteeinheiten festgelegt
MARKER_ART (M_ART)	Markerattribut - es können 28 Markerarten festgelegt werden Beispiele: M_ART Markersymbol 1 . 2 + 3 * 4 0 5 x . . . . . .
LINIE (X1,Y1,X2,Y2)	Die Punkte P1(X1,Y1) und P2(X2,Y2) werden gemäß der Linienattribute miteinander verbunden. X1,Y1,X2,Y2 : <0..32767>
LINIENSTAERKE (STAERKE)	Linienattribut (7 Stufungen möglich); SRAERKE wird in Geräteeinheiten angegeben.
LINIE_TYP (TYP)	Linienattribut - geräteabhängig, aber mindestens 5 TYP : 1 - Vollinie 2 - Strichlinie 3 - Punktlinie 4 - Strich-Punkt-Linie 5 - Strich-Punkt-Punkt-Linie
KREIS (XM,YM,RADIUS)	Vollkreis mit Mittelpunkt (XM,YM) und RADIUS; (gemäß Füllart und Füllindex) XM,YM,RADIUS : <0..32767)
BALKEN (X1,Y1,X2,Y2)	Rechteck mit der linken unteren Ecke (X1,Y1) und der rechten oberen Ecke (X2,Y2); (gemäß Füllart und Füllindex) X1,Y1,X2,Y2 : <0..32767>
KREISBOGEN (XM,YM,RADIUS, AW,EW)	Kreisbogen mit Mittelpunkt (XM,YM) und RADIUS, sowie Anfangswinkel (AW) und Endwinkel (EW); XM,YM,RADIUS : <0..32767> AW,EW : <0..3600> - in Zehntelgraden
FUELL_ART (F_ART)	Festlegen der Füllart F_ART : 0 - leer 1 - voll 2 - gemustert 3 - schraffiert
FUELLINDEX (INDEX)	Festlegen des Füllindex (abhängig von der verwendeten Füllart); Der Füllindex verweist auf eine Muster- bzw. Schraffurart (bei F_ART=3 bzw. 4). INDEX : <1..geräteabhangiges Maximum>
PUNKT (X,Y)	Zeichnen des Punktes mit den Koordinaten X und Y. X,Y : <0..32767>
TEXTAUSGABE (X,Y,TEXT)	Der Inhalt von TEXT wird an der Position X,Y ausgegeben. TEXT : STRING(.40.) X,Y : <0..32767>
EINGABE_LOCATOR (X,Y)	Die Locatorkoordinaten (z.B.: Fadenkreuz) werden in Gerätekoordinaten in X und Y übergeben. Das Fadenkreuz läßt sich folgendermaßen in Verbindung mit der CONTROL - Taste bewegen: ^ = Bildmitte ^ -> nach rechts ^ <- nach links ^ ↑ nach oben ^ ↓ nach unten ^ 2 nach rechts oben ^ - nach rechts unten ^ 1 nach links oben ^ 0 nach links unten Abschluß der Locatoreingabe: ^ 9 ohne Wertübergabe ^ ENT mit Wertübergabe Außerdem kann die Splitgrenze zwischen Alphanumerik- und Grafikbild verändert werden: ^ : Verschieben der Splitgrenze nach oben ^ * Verschieben der Splitgrenze nach unten
SPLITGRENZE (ZEILE)	Festlegen der Splitgrenze auf eine Zeile im Bereich 1..25. Bis ZEILE wird das Grafikbild dargestellt und ab ZEILE+1 das Alphanumerikbild.

```
PROGRAM EDI;
  VAR X,Y,X1,Y1,X2,Y2: INTEGER;
      A,B             : BOOLEAN;
      TXT             : STRING[40];

{$ITURBOGX.PAS}

BEGIN
  GRAFIK_EIN (BILDSCHIRM);
  X1:=3000;
  Y1:=0;
  X:=2000;
  LINIE (X1,X,X1,Y1);          { Umrahmen des ENDE-Feldes }
  LINIE (Y1,X,X1,X);
  TXT:='  ENDE';
  TEXTAUSGABE (Y1,Y1,TXT);     { Beschriften des ENDE-Feldes }
  EINGABE_LOCATOR (X,Y);       { Laden von Anfangswerten }
  X1:=X;
  Y1:=Y;
  A:=TRUE;
  B:=TRUE;
  WHILE (A OR B) DO BEGIN      { Editieren, solange der Locator      }
    EINGABE_LOCATOR(X,Y);      { nicht im ENDE-Feld ausgeloest wird }
    A:=X>3000;
    B:=Y>2000;
    X2:=X1;
    Y2:=Y1;
    X1:=X;
    Y1:=Y;
    IF (A OR B) = TRUE THEN    { Der Linienzug in das ENDE-Feld }
    BEGIN                      { wird nicht gezeichnet          }
      LINIE (X2,Y2,X1,Y1);
    END;
   END;
  HARDCOPY;                    { Ergebnisausdruck }
  GRAFIK_AUS;
END.
```

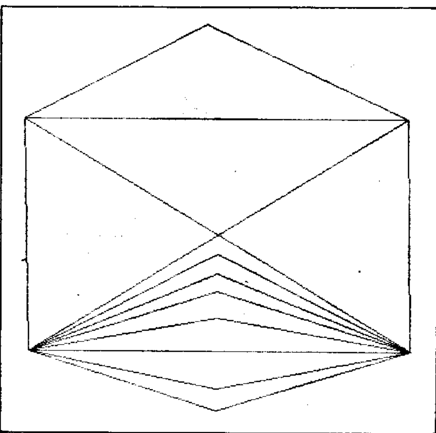


Bild 2a Linieneditor
Bild 2b Ergebnis eines Editiervorganges

Dipl.-Ing. Ottmar Vetter (30) studierte von 1978 bis 1983 an der Technischen Universität Dresden, Sektion Elektrotechnik. Von 1983–1987 arbeitete er im VEB Robotron-Elektronik Dresden, vor allem an der Entwicklung des A 7100. Seit diesem Jahr ist er im VEB Elektroprojekt und Anlagenbau Berlin tätig und befaßt sich zur Zeit mit Problemen der digitalen Simulation von Elektroantrieben auf 16-Bit-Rechentechnik.

```
PROGRAM FUNKTION;

VAR     L,M,X,Y  : REAL;
        N,M_ART,K1,K2,K3,K4,K5,K6,K7,K8: INTEGER;
        TXT          : STRING[40];

{$ITURBOGX.PAS}

BEGIN
  GRAFIK_EIN (BILDSCHIRM);
  K1:=0;                       { Koordinatenfestlegungen }
  K2:=16200;
  K3:=29000;
  K4:=30000;
  K5:=500;
  K6:=16000;
  M:=100;
  LINIE(K1,K2,K3,K2);          { X - Achse }
  LINIE(K5,K1,K5,K4);          { Y - Achse }
  M_ART:=28;
  MARKER_ART(M_ART);           { Y - Pfeil }
  MARKER(K5,K4);               { Pfeil zeichnen }
  M_ART:=26;
  MARKER_ART(M_ART);           { X - Pfeil }
  MARKER(K3,K2);               { Pfeil zeichnen }
  TXT:='Y Achse';
  K1:=K1+1000;
  TEXTAUSGABE(K1,K3,TXT);      { Beschriftung der Y-Achse }
  TXT:='X Achse';
  K2:=14500;
  K3:=26500;
  TEXTAUSGABE(K3,K2,TXT);      { Beschriftung der X-Achse }
  TXT:='Sinusfunktion';
  K3:=20000;
  K2:=25000;
  TEXTAUSGABE(K3,K2,TXT);      { Funktionsbeschriftung }
  FOR N:=0 TO 200 DO
    BEGIN                      { Berechnen und Zeichnen }
      L:=ROUND(N);             { der Sinusfunktion      }
      X:=(L*PI)/M;
      Y:=SIN(X);
      K7:=K5;
      K5:=ROUND((L*13000/M)+500);
      K8:=K6;
      K6:=ROUND((10000*Y)+16200);
      LINIE(K5,K6,K7,K8);
    END;
  HARDCOPY;
  REPEAT UNTIL KEYPRESSED;     { Programmende bei Tastendruck }
  GRAFIK_AUS;
  END.
```

SCP-GX Steuercodes übergeben oder Rückinformationen von SCP-GX enthalten. Die an SCP-GX übergebenen Steuercodes werden interpretiert und lösen die jeweilige Grafikfunktion aus.

4. TURBO-PASCAL ruft SCP-GX

Um SCP-Systemrufe zu ermöglichen, ist in TURBO-PASCAL die Prozedur
'BDOS (Parameter)'
eingeführt worden.
Die Parameter sind vom Typ:
RECORD
AX, BX, CX, DX, BP, SI, DI, DS, ES, FLAGS: INTEGER; END;
Die für die SCP-GX notwendige Parameterliste wird durch Felder realisiert. Nun ist man in der Lage, ein PASCAL-Quellprogramm zur Definition von Grafikprozeduren zu erstellen. Diese Quelle bindet man in sein Anwenderprogramm als Include-Datai ein.

5. Realisierte Grafikprozeduren

Im folgenden werden kurz die realisierten grafischen Ein- und Ausgabeprozeduren vor-

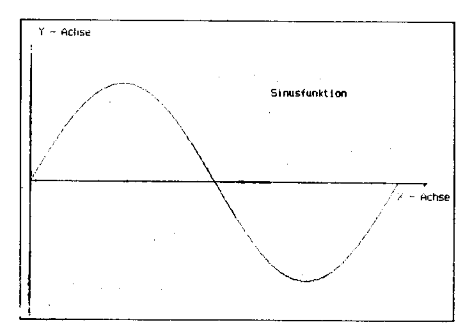


Bild 3a Programm zum Zeichnen einer Sinusfunktion
Bild 3b Hardcopy eines Programmlaufes

```
PROGRAM PROCEDURTEST;

TYPE       TSTRING =  STRING[40];

VAR        F_ART,IND,MART,RAD1,RAD2,A,B,
           AW,EW,X1,X2,Y1,Y2,X3,Y3          :INTEGER;
           TXT :TSTRING;

{$ITURBOGX.PAS}

BEGIN
   X1:=3000;                    { Koordinatenfestlegungen }
   Y1:=3000;
   X2:=10000;
   Y2:=25000;
   X3:=20000;
   Y3:=20000;
   CLRSCR;

{ Parametereingaben }

   WRITE('  LINIENSTAERKE  1..500  :  ');
   READLN(A);
   WRITE('  Eingabe eines kurzen Textes :  ');
   READLN(TXT);
   WRITE('  RADIUS fuer Kreis  ( 1..20000 ) :  ');
   READLN(RAD1);
   WRITE('  RADIUS fuer Kreisbögen ( 1..20000 ) :  ');
   READLN(RAD2);
   WRITE('  Anfangswinkel fuer Kreisbogen ( 0..3600 ) :  ');
   READLN(AW);
   WRITE('  Endwinkel fuer Kreisbogen ( 0..3600 ) :  ');
   READLN(EW);
   WRITE('  FUELLINDEX  :  ');
   READLN(IND);
   WRITE('  FUELLART  :  ');
   READLN(F_ART);
   WRITE('  MARKERART  :  ');
   READLN(MART);

{ Grafikbilddarstellung gemaess der Parametereingaben }

   GRAFIK_EIN(BILDSCHIRM);
   CLR_GRAFIK;
     LINIENSTAERKE(A,B);
     FUELL_ART(F_ART);
     FUELLINDEX(IND);
     MARKER_ART(MART);
     LINIE(X1,Y1,X3,Y2);
     KREIS(X2,Y2,RAD1);
     BALKEN(X1,Y1,X2,Y2);
     MARKER(X3,Y3);
     Y3:=Y3-3000;
     TEXTAUSGABE(X3,Y3,TXT);
     KREISBOGEN(X3,Y3,RAD2,AW,EW);
   REPEAT  UNTIL KEYPRESSED;          { Abbruch bei Tastendruck }
   GRAFIK_AUS;
END.
```

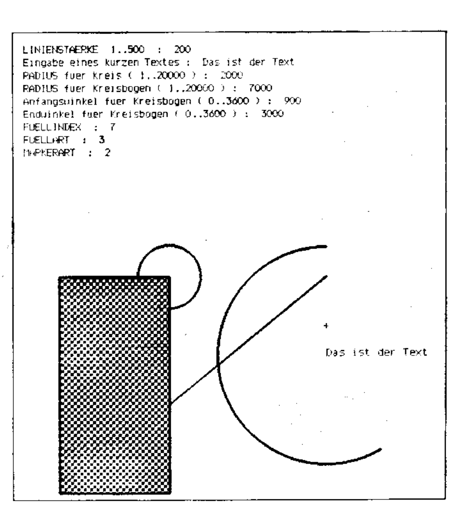


Bild 4a Testprogramm für einige Grafikprozeduren
Bild 4b Ergebnis der Testroutine

gestellt. Als grafisches Eingabemedium dient die Tastatur, und grafische Ausgaben können wahlweise auf Bildschirm, Plotter oder Drucker weitergeleitet werden. Das Anwenderprogramm übergibt die grafischen Koordinaten für die X- und Y-Achse jeweils als Normalisierte Gerätekoordination von 0 bis 32767.
GDOS wandelt diese Koordinatenangaben in Gerätekoordinaten für das spezielle Grafikgerät um.
In Bild 1 sind die wichtigsten Grafikprozeduren, welche in der Include-Datei TURBOGX.PAS enthalten sind, aufgelistet. Es ist zu beachten, daß bestimmte Prozeduren geräteabhängige Besonderheiten aufweisen /2/.

6. Beispiele

Als Demonstration zur Handhabung der Prozeduren von TURBOGX.PAS sollen die folgenden kleinen Beispiele dienen. In Bild 2a ist der Quelltext eines einfachen Linieneditors dargestellt. Das Editieren erfolgt gemäß der Prozedurbeschreibung von EINGABE_LOCATOR (Bild 1). Wird der Locator im ENDE-Feld abgeschlossen, erfolgt ein Ergebnisausdruck des Editiervorganges.
Bild 2b zeigt ein mögliches Ergebnis des Editiervorganges.
Das Programm von Bild 3a bildet eine Periode der Sinusfunktion auf dem Bildschirm ab und erzeugt anschließend eine Hardcopy (Bild 3b).
Bild 4a enthält ein Testprogramm für einige Grafikprozeduren, und Bild 4b veranschaulicht die Arbeitsweise der Testroutine.

Literatur

/1/ S. Kerst, H. Riegel: Arbeitsplatzcomputer robotron A 7100. edv-aspekte 1/1987
/2/ U. Heckel: Das Betriebssystem SCP 1700. edv-aspekte 1/1987

✉ KONTAKT ☏

VEB Elektroprojekt und Anlagenbau Berlin, Sitz Technische Universität Dresden, Helmholtzstr. 9, Dresden, 8027; Tel.: 46 34 664

Grafikprogramm für den KC 85/3

Der große Vorteil des KC 85/3 liegt in der vorhandenen Vollgrafik, welche in BASIC leicht nutzbar ist. In vielen Fällen stört jedoch die – auch vom Programm abhängige – langsame Zeichengeschwindigkeit. Aus diesem Grund wurde das Maschinenprogramm DRAWER entwickelt. Es hat einen Umfang von 4 KByte und liegt im Speicher von 7000H bis 7FFFH. Ein frei wählbarer Bildspeicher liegt auf den Adressen 5000H bis 6FFFH. Es besteht die Möglichkeit, das Programm sowohl von CAOS aus mittels DRAW oder REDRAW (letzter Befehl löscht nicht das Schirmbild) als auch von jedem BASIC-Programm aus mit CALL*7001 zu starten. Das Programm ermöglicht die Bearbeitung jeglicher Grafik auf dem Bildschirm; dazu wird dieser aufgeteilt in ein 256×256 großes Pixelfeld und eine 8×32 Zeichen große Spalte für die Anzeige des Menüs, von Informationen, Koordinaten u. ä. Das Programm besteht aus den sechs Befehlsgruppen Steuerbefehle, Kassettenbefehle, Speicherbefehle, logische Bildbefehle, Bildbefehle und Zeichenbefehle.
Die Bildbefehle z. B. realisieren ein in den Grenzen frei wählbares Verschieben, Spiegeln, Löschen, Beschriften und Einfärben des Bildes, während die sechste Gruppe das Zeichnen von Linien, Kreisen und Ellipsen entsprechend eingegebener Koordinaten sowie als Kernstück die sogenannte Turtlegrafik umfaßt. Über ein zweites Menü können unter anderem alle 11 Options der Turtlegrafik (z. B. das Ausfüllen von Flächen) genutzt werden. Das gesamte Programm ist gegen Falscheingaben gesichert. Es ist universell sowohl als selbständiges Programm als auch als Modul für BASIC geeignet.

EOS „Dr. T. Neubauer", Schloßplatz, Arnstadt, 5210 *Eckoldt*

Grafikprogramm zur Darstellung von Ergebnissen der Aufwärtsübersetzung

Gerald Tränkner
Ingenieurhochschule Mittweida,
Sektion Informationselektronik

Das beschriebene Programmsystem GRAPH ist ein Grafikprogramm in Assembler und dient der Darstellung von Programmteilen in PAP-ähnlicher Form. Hardwaremäßig wird der Grafikmodul LC 8552 auf Basis von K-1520-Systemen genutzt. Die verwendeten Unterprogramme sind einzeln ansprechbar und können mit geringem Steueraufwand auch für andere Zwecke zusammengestellt werden. Die wesentlichsten Eigenschaften, Funktionsprinzipien und Strukturen werden vorgestellt. Bei der Erarbeitung des Programms wurde vor allem auf die Universalität und eine hohe Arbeitsgeschwindigkeit Wert gelegt.
Aus diesem Grund wurde zum Zeichnen nur auf waagerechte und senkrechte Linien zurückgegriffen. Mit geringem Steueraufwand sind allerdings auch Linien beliebiger Anstiegswinkel darstellbar.

Grundkonzeption

Im Gegensatz zu bekannten Programmen werden keine schrägen Verbindungen genutzt. Die Realisierung der geforderten Zeichengeschwindigkeit wird durch die Zeichnung von ausschließlich waagerechten und senkrechten Linien erreicht. Unterschiedliche Linienarten und -stärken dienen der Darstellung besonderer technischer Sachverhalte. Der gesamte Inhalt des Bildschirmes ist in einem speziellen Telegramm codiert enthalten. Dieses Telegramm wird in einem Host-Rechner generiert und vom Programm entschlüsselt. Die Software ist für eine Erweiterung vorgesehen und ermöglicht die Nutzung für größere Bildschirme. Durch den aufrüstbaren Telegramminhalt sind die Formate flexibel gestaltbar.

Bildaufbau

Möglichkeiten des Programms im einzelnen:

- Punkte lt. Koordinaten setzen und rücksetzen
- zwischen gegebenen Punktkoordinaten waagerechte und senkrechte Zeichnung von:
  - Linien dreifacher Stärke
  - einfachen Linien
  - Strichellinien
- Darstellung von Programmknotenpunkten als Rechtecke auf bestimmten, fest definierten Bildschirm-Flächen
- Erkennung bestimmter, markanter Übergabefehler im Telegramm
- Realisierung der Schrift im Format 7×5 Bit

Die darzustellenden Programmteile werden folgendermaßen auf den Bildschirm gebracht:

▪ Jeder Befehl wird entsprechend der Mnemonik ausgeschrieben und von „Balken" oberhalb und unterhalb der Schrift begrenzt – diese Darstellung wird Knoten genannt.

▪ Die Balken können in 4 verschiedenen Arten dargestellt werden, welche die Anzahl der Steuerflüsse beinhalten, die an diesen Balken angeschlossen sind.

▪ Im Programmablauf linear aufeinander folgende Befehle werden als senkrecht untereinander stehende Knoten beschrieben; bei Verzweigungen können bis zu 3 parallele Programmabläufe dargestellt werden.

▪ Die Verbindung der Knoten erfolgt mittels Steuer- und Datenflußlinien, welche als Dreifach- bzw. Strichel-Linien dargestellt werden.

▪ Ein Knoten wird durch die Steuer- und Datenflüsse angesprochen, die von oben an den oberen Balken anschließen und spricht andere Knoten durch Steuer- und Datenlinien an, die nach unten vom unteren Balken abgehen.

▪ Die freien senkrechten Streifen neben den Knoten werden als Verbindungskanäle bezeichnet und beinhalten Steuer- Datenflußlinien, die über mehr als 2 Waagerechtfelder hinausgehen.
Auf dem Bildschirm können waagerecht 3 Knoten nebeneinander und senkrecht 8 Knoten untereinander abgebildet werden (Bild 1).

Programmeinschränkungen

Falls durch einen Programmteil im mittleren Senkrechtfeld eine Verbindungslinie vom rechten zum linken Senkrechtfeld läuft, ist für diese keine gesonderte Darstellung vorgesehen.
Dadurch kann es zur Überlagerung von Linien kommen, welche zwar die Auswertung erschwert, aber nicht die Eindeutigkeit der Verbindung beeinträchtigt.
Ebenfalls aus Platzgründen wurde die Länge der eingeschriebenen Mnemonik auf 16 Buchstaben begrenzt. Was darüber hinaus geht, wird weggeschnitten. Des weiteren existiert kein Überschreibschutz des aufgebauten Bildes.

Inhaltliche Schwerpunkte

Das Programm GRAPH wird als Unterprogramm durch ein Hauptprogramm angesprochen und erhält von diesem die notwendigen Angaben über die zu zeichnenden Programmstücke. Die Daten werden vom Hauptprogramm in einem Telegramm formiert übergeben und von einer Startadresse (siehe Bild 2) beginnend im RAM abgelegt. Der Inhalt des Telegramms ist aus Bild 2 ersichtlich. Die Feld-Nummern sind 2 Byte lang, alle anderen Daten 1 Byte. Die Verwaltung des Telegramms, das heißt das Weiterzählen der Telegrammadressen, auf die zugegriffen wird, wurde vom Indexregister IY übertragen.
In Anpassung an die Telegrammgestaltung ergibt sich der Programmablauf von GRAPH laut Bild 3.
Zur Vermeidung von Überlagerungen von Linien im Verbindungskanal wurden dort Lauf-Koordinaten in x-Richtung definiert, die nach jedem Linien-Zug geändert werden.
Um auftretende markante Telegrammübertragungsfehler zu registrieren, ist im RAM ein spezielles Fehlerregister-Byte reserviert, welches je nach Fehlerart geladen wird. Der Inhalt dieser Speicherstelle wird nach erfolgtem Bildaufbau ausgewertet und als Fehlermeldung mit auf den Bildschirm gebracht. Feld-Nummern, die außerhalb des Bildschirmes liegen, initialisieren zwar eine Fehlerausschrift, unterliegen aber ansonsten dem normalen Programmablauf. Das heißt, die auf dem Bildschirm sichtbaren Steuer- und Datenlinien werden so weit gezeichnet, als ob die gezeigte Struktur einen Ausschnitt einer noch größeren Übersicht darstellt.
Die 8 Waagerecht- und 3 Senkrechtfelder des Bildschirmes ergeben 24 Felder, welche wie Matrizen nach Zeile und Spalte bezeichnet sind. Für jedes Feld wurden des weiteren noch eine Startadresse (für Zeichnung des Kastens) und xo, yo Koordinaten definiert (für Einschreiben der Mnemonik).
Eine möglichst hohe Arbeitsgeschwindigkeit der Programme wurde u. a. durch die Ver-

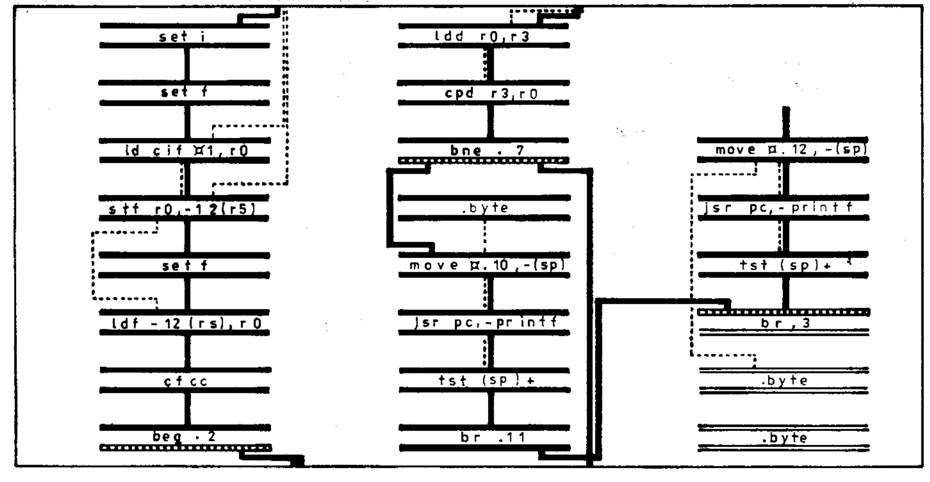


Bild 1 Beispiel eines Bildaufbaus

Telegrammstartadresse	M	Anzahl der Knoten
	Start-Feld-Nr. 1	Feld-Nr. des 1. Knotens
	N1	Anzahl der Befehlszeichen
	Mnemonic	N-mal ASCII-Daten der zu schreibenden Befehle
	T0	Teilung des oberen Balkens
	TU	Teilung des unteren Balkens
	i VST	Anzahl der Vorgänger-Steuerflüsse
M-mal	Ziel-Feld-Nr. … Ziel-Feld-Nr.	i-mal Feld-Nrn. der Knoten der Vorgänger-Steuerlinie
	j NST	Anzahl der Nachfolger-Steuerflüsse
	Ziel-Feld-Nr. … Ziel-Feld-Nr.	j-mal Feld-Nrn. der Knoten der Nachfolger-Steuerlinie
	k VD	Anzahl der Vorgänger-Datenflüsse
	Ziel-Feld-Nr. … Ziel-Feld-Nr.	k-mal Feld-Nrn. der Knoten der Vorgängerdaten
	l ND	Anzahl der Nachfolger-Datenflüsse
	Ziel-Feld-Nr. … Ziel-Feld-Nr.	l-mal Feld-Nrn. der Knoten der Nachfolgerdaten
	Start-Feld-Nr. 2	
	N2	
	Mnemonic	

Bild 2
Telegrammaufbau

- Bildschirm löschen
- Laufkoordinaten rücksetzen
- Anfangsadresse des Bildschirmwiederholspeichers laden
- (Fehlerregistrieradresse) auf 00 setzen
- IY mit Telegrammstartadresse laden
- M-mal
  - Start-Feld-Nr. übernehmen
  - Feld-Koordinaten ermitteln
  - N übernehmen = Anzahl der Buchstaben
  - Anfangskoordinaten für Mnemonic ermitteln
  - Mnemonic einschreiben
  - T0 übernehmen und oberen Balken zeichnen
  - TU übernehmen und unteren Balken zeichnen
  - i übernehmen
  - i-mal
    - Ziel-Feld-Nr. übernehmen
    - Gleiche Spalte? – ja: Verbindung mit Steuerlinie nach oben – nein: in Richtung Ziel-Feld Verbindung mit Steuerlinie
  - j übernehmen
  - j-mal
    - Ziel-Feld-Nr. übernehmen
    - Gleiche Spalte? – ja: Verbindung mit Steuerlinie nach unten – nein: in Richtung Ziel-Feld Verbindung mit Steuerlinie
  - k übernehmen
  - k-mal
    - Ziel-Feld-Nr. übernehmen
    - Gleiche Spalte? – ja: Verbindung mit Datenlinie nach oben – nein: in Richtung Ziel-Feld Verbindung mit Datenlinie
  - l übernehmen
  - l-mal
    - Ziel-Feld-Nr. übernehmen
    - Gleiche Spalte? – ja: Verbindung mit Datenlinie nach unten – nein: in Richtung Ziel-Feld Verbindung mit Datenlinie
- Inhalt der Fehlerregistrieradresse im Fehler-UP auswerten
- RETURN

Bild 3
Struktogramm des Programms

wendung von Bibliotheken erreicht. Zur Nutzung des Programms in der vorhandenen Form müssen die Daten dem Telegramminhalt angepaßt und als solcher im RAM abgelegt werden. Will man nur Programmteile nutzen oder andere Anforderungen erfüllen, so kann man mit vertretbarem Steueraufwand auf die einzelnen Unterprogramme zurückgreifen.

Programmablauf

Um einen Programmknoten und seine Verbindungen zu anderen Knoten darzustellen, sind folgende Schritte notwendig:

① Mit Übernahme der Start-Feld-Nummer wird dem Knoten ein „Bereich" auf dem Bildschirm zugewiesen, daraus können x,y-Koordinaten und eine bestimmte Adresse errechnet werden.

② Die nächste aus dem Telegramm entnommene Information enthält die Anzahl N der Befehlszeichen und gewährleistet somit ein Einrücken der Mnemonik in die Mitte des Knotens.

③ Diese wird entsprechend den folgenden ASCII-Codes eingeschrieben.

④ Mit Hilfe der nächsten Telegrammbytes werden zuerst der obere und dann der untere Balken entsprechend der Anzahl der anliegenden Steuerflüsse zur Darstellung gebracht.

⑤ Die folgenden Daten enthalten zunächst die Anzahl und dann die Nummern der Ziel-Felder, von denen ausgehend Steuerflüsse an den soeben gezeichneten oberen Balken führen. Dabei werden die senkrechten Linien im Verbindungskanal immer so neben bereits vorhandenen vorbeigeführt, daß keine Überlappung auftreten.

⑥ Nach vollständiger Darstellung der Vorgängersteuerflüsse wiederholt sich der Prozeß für die Nachfolgersteuerflüsse mit dem Unterschied, daß die Linien, vom unteren Balken ausgehend, an den oberen des Ziel-Feldes gezogen werden.

⑦ Analog zu den Steuerflüssen werden die Vorgänger- und Nachfolgerdatenflüsse dargestellt, allerdings werden anstatt Dreifachlinien nun Strichellinien verwendet.

Danach beginnt diese Folge von neuem solange, bis alle M Programmknoten dargestellt sind. Zur Unterstützung der Arbeit erfolgt bei bestimmten Fehlern eine entsprechende Ausschrift in der rechten unteren Ecke des Bildschirmes.

Literatur

/1/ Kinder, F.; Schufft, M.: Bildschirmsteuerung für K 1520. Radio, Ferns., Elektron., Berlin 35 (1986) 10, S. 627–629

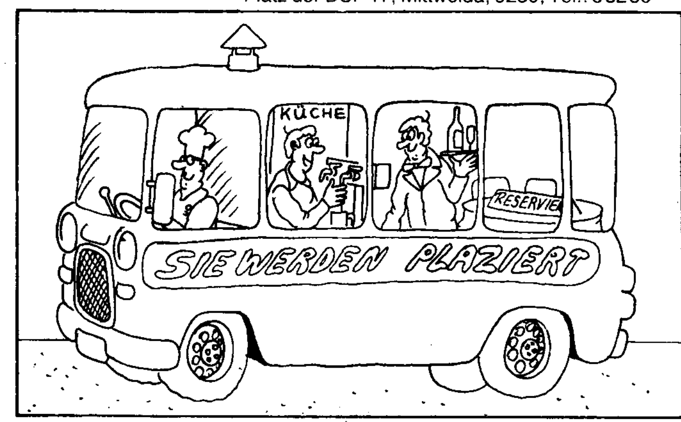
/2/ Eine Grund-Software für alphanumerische und graphische Darstellungen auf dem Rasterfarbdisplay FD 4996. W. Schmiedel, 1984, ZfK Rossendorf

/3/ Dokumentation zum Graphikmodul LC 8552. IHM ZEG, Abt. Konstruktion, 1986

/4/ Aufbau eines Graphikdisplays für den K 1520. Nonn, Diplomarbeit IHM 1983

✉ KONTAKT ☏

Ingenieurhochschule Mittweida, SG 84.11.04, Platz der DSF 17, Mittweida, 9250; Tel.: 58286



Lokal-Bus
Zeichnung: Frank Steger

Druckergrafik für technische Anwendungen

Christian Hanisch
Technische Universität Dresden,
Sektion Wasserwesen

Wie PC und BC, die über keine hardwareseitigen Voraussetzungen für Bildschirmgrafik verfügen, dennoch an der großen Palette der Computergrafik teilhaben können, soll nachfolgend an einem Praxisbeispiel aus der Montanhydrologie gezeigt werden. Voraussetzung ist lediglich ein grafikfähiger Matrixdrucker an besagten Geräten.

Im Hauptspeicher des Rechners wird dazu ein angemessen großer Bereich als virtueller Grafikbildschirm reserviert. Eine *Auflösung von 640×200* oder 480×320 *Punkten* stellt dabei einen guten Kompromiß zwischen der Größe des gedruckten Grafikbildes, der zur Generierung von Grafikelementen auf diesem Pseudodisplay benötigten Rechenzeit und des noch für das Nutzerprogramm verfügbaren Speicherplatzes in einem 64-KByte-Hauptspeicher dar. Die angegebene Grafikauflösung (640×200) entspricht darüber hinaus international üblichen Größenordnungen für Bildschirmgrafik.
Die Verwaltung des Pseudodisplays erfolgt in Richtung x-Achse byteweise und in y-Richtung bitweise. Damit wird ein 640 × (200/8) = *16 000 Byte* oder 480 × (320/8) = 19 200 Byte großer ARRAY im Hauptspeicher benötigt. Die hier vorzustellende Prozedur zum Schraffieren einer Fläche baut auf Elementarroutinen (Setzen von Punkten, Ziehen von Linien usw.) auf, die in ähnlicher Form auch in /1/ und /2/ dargestellt sind. Zur Steigerung der Leistungsfähigkeit des Routinenpaketes wäre das Ersetzen des sehr einfachen Linienalgorithmus (Prozedur LINE) durch einen BRESENHAM-Linienalgorithmus denkbar /3/.
Bei dem als Beispiel dienenden Anwendungsfall (Bild 1) werden die Geometrie und der Ausgangszustand eines Systems für ein numerisches Simulationsmodell dargestellt. Das Modell dient zur Berechnung der Wirkungen von Tagebauentwässerungsanlagen. Es war erforderlich, die im Modellschnitt unterschiedlichen Schichten (Grundwasserleiterstauer, Grundwasserleiter) optisch gegeneinander abzugrenzen. Dabei lag die Anwendung von Schraffuren für Grundwasserleiterstauer sowie verschiedenen Linientypen zur Darstellung der Ausgangswasserstände nahe. Die zur Entwässerung dienenden Brunnengalerien sind im Schnitt als Stäbchen dargestellt. Die HiRes-Grafik dient Dokumentationszwecken und stellt darüber hinaus ein Mittel zur visuellen Prüfung der geometriebeschreibenden Eingangsdaten dar.
Die für den beschriebenen Anwendungsfall verwendete Schraffurroutine (FILLAREA, Bild 2) gestattet eine variable Schraffurrichtung und -dichte. Der zu schraffierende Bereich wird durch einen Polygonzug begrenzt, der ihn in mathematisch positivem Sinne umläuft. Die Punkte dieses Polygonzuges werden der Prozedur durch zwei Vektoren übergeben, die entsprechende x- und y-Koordinaten enthalten. Der Typ VEKTOR muß daher im Hauptprogramm als ein REAL-ARRAY vereinbart sein. Gegebenenfalls muß in einer vorgelagerten Prozedur dafür gesorgt werden, daß die Koordinaten der Punkte in die Größenordnung des Grafikbildschirmes (z. B. 640×200) transformiert werden. Die Prozedur FILLAREA transformiert zunächst die ihr übergebenen y-Koordinaten in ein Koordinatensystem, dessen x-Achse parallel zur Schraffurrichtung liegt. Diese neuen y-Koordinaten liefern durch ihren kleinsten bzw. größten Wert Anfangs- und Endpunkt des Bereiches, in dem die Schraffurgeraden liegen. Dann werden alle Schnittpunkte des Polygonzuges mit der jeweils betrachteten Schraffurgeraden ermittelt. Diese werden in einem sogenannten Binärbaum geordnet. Anfangs- und Endpunkt einer Schraffurlinie liefern nun der erste und zweite, dritte und vierte usw. Punkt des Binärbaumes, da die Schnittpunkte stets paarig auftreten müssen. Um dies unter allen Umständen zu gewährleisten, wurde eine Abbruchschranke zum Ausschließen von Rundungsfehlern programmiert. Weiterhin ist eine Korrektur des Versatzes (Abstand zur Nebengeraden) der Schraffurlinie vorgesehen, falls ein Eckpunkt des Polygons als Schnittpunkt erkannt wird.
Die aufgeführten Routinen können über die INCLUDE-Option des TURBO-PASCAL-Compilers in den Nutzerquelltext eingebunden werden (siehe Bild 3).
Sie stellen bereits ein leistungsstarkes Grundpaket dar. Im dargestellten Beispiel (Bild 1) wurden lediglich noch eine Prozedur für gestrichelte Linien, eine Prozedur zum Löschen von Punkten und eine DUMP-Prozedur verwendet, die um ein bildschirmähnliches Aussehen der Grafik zu erzeugen, alle Pixel in y-Richtung doppelt /1/.
Das Mischen von Grafik und Text kann auf einfache Weise erfolgen, indem der Grafikbildschirm mit einem Textbildschirm überlagert ausgedruckt wird. Als Textbildschirm kann dabei der Bildschirm des PC/BC selbst oder auch ein virtueller Speicherbereich in der Größe des alphanumerischen Bildschirmes dienen, der mit den gewünschten Textinformationen über Prozeduren versorgt wird. /1/ Die Rechenzeit zur Erstellung der als Bild 1 angefügten HiRes-Grafik betrug 13 Minuten auf PC 1715.

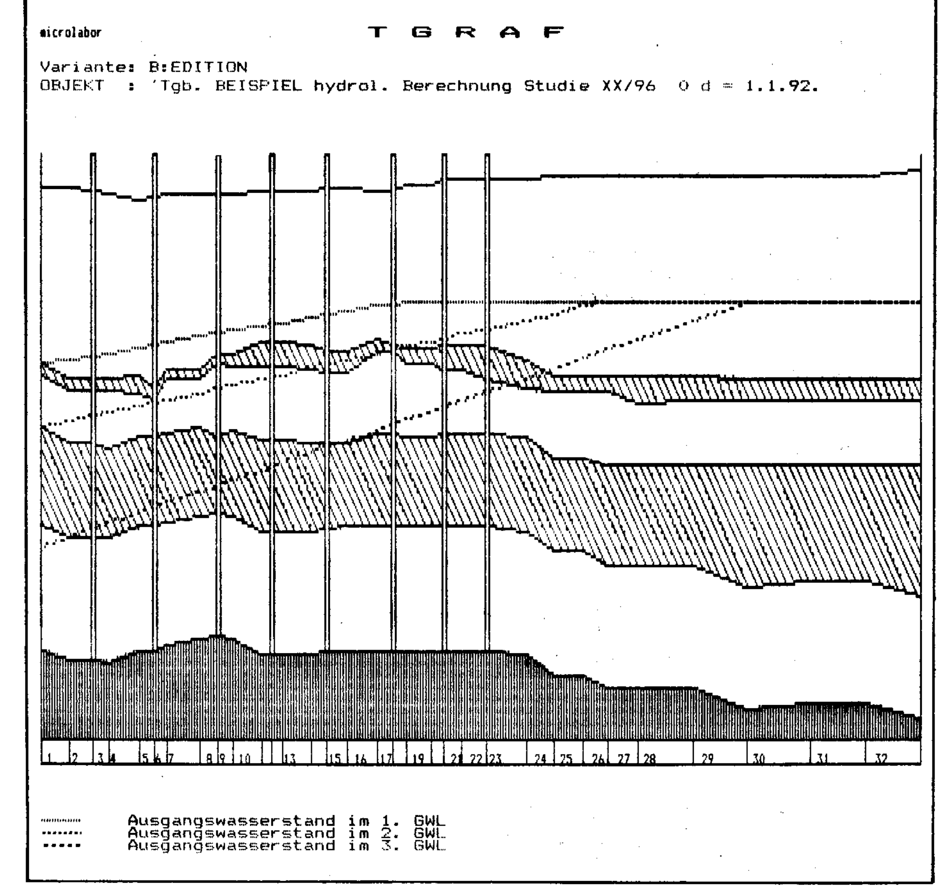


Bild 1 Praktisches Anwendungsbeispiel des Grafikpaketes

Literatur

/1/ Grafik ohne Bildschirm. CHIP-Sonderheft „TURBO-PASCAL", Vogel-Verlag, Würzburg
/2/ Plate, J.: Hochauflösende Drucker-Grafik. mc, Franzis-Verlag, München (1986) 9, S. 62–65
/3/ Cramer, E.: Erweiterter Bresenham-Algorithmus. mc, Franzis-Verlag, München (1987) 4, S. 66, 67

KONTAKT

Technische Universität Dresden, Sektion Wasserwesen, Mommsenstraße 13, Dresden, 8027; Tel. 232 61 18

```
0 CONST xmax=639; (* Grafikaufloesung 640 x 200 Punkte *)
0       ymax=199; (* 200/8=25 Byte vertikal            *)
0
0 VAR  pb : ARRAY[0..24,0..xmax] OF BYTE;
0
0 (*----------------------------------------------------*)
0 PROCEDURE CLRPB;
0 (* loescht Grafikbildschirm, ist am Beginn aufzurufen *)
0 BEGIN fillchar(pb,sizeof(pb),#0) END;
0
0 (*----------------------------------------------------*)
0 PROCEDURE SETDOT(x,y:INTEGER);
0 (* setzt einen Punkt *)
0 VAR h1,h2 : INTEGER;
0 BEGIN y:=ymax-y;
1   IF (0<=x) AND (x<=xmax) AND (0<=y) AND (y<=ymax) THEN
1   BEGIN
2     h1:=y SHR 3; h2:=y MOD 8;
2     pb[h1,x]:=pb[h1,x] OR (128 SHR h2)
2   END (* dagegen: Entfernen eines Punktes mit
1             pb[h1,x]:=pb[h1,x] and not(128 shr h2)    *)
1 END;  (* SETDOT *)
0
0 (*----------------------------------------------------*)
0 PROCEDURE LINE(ax,ay,bx,by:INTEGER);
0 (* zieht eine Linie von ax,ay nach bx,by *)
0 VAR g,i,h1,h2       : INTEGER;
0     dx,dy,l,ds,dc: REAL;
0 BEGIN
1   dx:=bx-ax; dy:=by-ay;
1   l:=sqrt(sqr(dx)+sqr(dy));
1   IF l=0 THEN setdot(ax,ay)
1          ELSE BEGIN
2                 ds:=dy/l; dc:=dx/l; g:=round(l);
2                 FOR i:=0 TO g DO BEGIN
3                   h1:=round(i*dc)+ax;
3                   h2:=round(i*ds)+ay;
3                   setdot(h1,h2)
3                 END
2               END
1 END;  (* LINE *)
0
0 (*----------------------------------------------------*)
0 PROCEDURE DUMP;
0 (* gibt den Grafikbildschirm auf dem Nadeldrucker aus *)
0 VAR i,j:INTEGER;
0 BEGIN
1   FOR i:=0 TO 24 DO BEGIN
2     (* nachfolgende Sequenzen fuer epson lx-86 *)
2     WRITE(lst,#27'3'#24);WRITE(lst,#27'<');
2     WRITE(lst,#27'*'#4,chr((xmax+1) MOD 256),
2                        chr((xmax+1) DIV 256));
2     FOR j:=0 TO xmax DO WRITE(lst,chr(pb[i,j]));
2     WRITELN(lst)
2   END;
1   WRITELN(lst,#27'2'#27'P')
1 END;    (* DUMP *)
0 (*----------------------------------------------------*)

0
0 PROCEDURE FILLAREA(x,y:vektor;fi:REAL;n,dichte:INTEGER);
0 (*******************************************************)
0 (* Diese Prozedur setzt einen Typ VEKTOR=array[1..n] OF   *)
0 (* REAL voraus!! Die Parameter x und y enthalten die      *)
0 (* korrespondierenden Punktkoordinaten des POLYGONZUG.    *)
0 (* Die zu schraffierende Flaeche wird dabei in            *)
0 (* mathematisch positiver Richtung umfahren.              *)
0 (* fi      - Schraffurwinkel in Bogenmass                 *)
0 (* n       - Anzahl der Punkte des Polygons               *)
0 (* dichte - Schraffurdichte Minimum=1                     *)
0 (*******************************************************)
0
0 LABEL m1,m2;
0
0 TYPE pointlist = ^point;
0        point   = RECORD
1                    xk,yk :REAL;
1                    l,r   :pointlist
1                  END;
0
0 VAR clipp                          : pointlist;
0     heapbottom                     : ^integer;
0     lax,lay,i,j                    : INTEGER;
0     distance,yanf,yend,xs,ys,s,
0     m,na,betragx,betragy,md,a1,a2,
0     nb,temp                        : REAL;
0     flag,flag1                     : BOOLEAN;
0
0 PROCEDURE putin(x,y:REAL);
0 VAR find:BOOLEAN; p,q :pointlist;
0
0 PROCEDURE create(VAR b: pointlist; x,y:REAL);
0   BEGIN
1     new(b);find:=TRUE;
1     WITH b^ DO BEGIN
2         l:=NIL; r:=NIL; xk:=x; yk:=y;
2     END
1   END;  (* create *)
0
0 BEGIN
1   find:=FALSE; q:=clipp;
1   IF clipp=NIL THEN create(clipp,x,y)
1     ELSE REPEAT IF x<q^.xk THEN IF q^.l=NIL THEN BEGIN
3                                                create(p,x,y);
3                                                q^.l:=p END
2                                              ELSE q:=q^.l
2                                   ELSE IF q^.r=NIL THEN BEGIN
3                                                create(p,x,y);
3                                                q^.r:=p END
2                                              ELSE q:=q^.r
2         UNTIL find;
1 END;  (* putin *)
0
0 {$A-}
0 PROCEDURE drawline(b:pointlist);
0 BEGIN
1   IF b<>NIL THEN BEGIN IF b^.l<>NIL THEN drawline(b^.l);
2     IF flag THEN BEGIN
3                    lax:=round(b^.xk);lay:=round(b^.yk);
3                    flag:=FALSE  END
2                    ELSE BEGIN
3       flag:=TRUE;
3       IF flag1 THEN line(lay,lax,round(b^.yk),round(b^.xk))
3                ELSE line(lax,lay,round(b^.xk),round(b^.yk))
3                  END;
2     IF b^.r<>NIL THEN drawline(b^.r)
2   END
1 END;{$A+}  (* drawline *)
0
0 BEGIN
1   mark(heapbottom);
1   yanf:=1E+36;yend:=-1E+36;flag1:=FALSE;
1   IF (dichte<1) OR (dichte>199) THEN dichte:=1;
1   IF fi>=pi THEN fi:=frac(fi/pi)*pi;
1   IF fi=pi*0.5 THEN BEGIN flag1:=TRUE;fi:=0 END;
1   FOR i:=1 TO n DO BEGIN
2     IF flag1 THEN BEGIN temp:=x[i];x[i]:=y[i];y[i]:=temp END;
2     (*  Koordinatentransformation  *)
2     distance:=cos(fi)*y[i]-sin(fi)*x[i];
2     IF distance<yanf THEN yanf:=distance;
2     IF distance>yend THEN yend:=distance
2   END;
1   md:=sin(fi)/cos(fi);
1 m2: WHILE yanf<=(yend-0.01) DO BEGIN
2     clipp:=NIL;
2     yanf:=yanf+dichte;
2     (* Ruecktransformation abs. Glied *)
2     nb:=(cos(-fi)-md*sin(-fi))*yanf; i:=1;
2     FOR i:=1 TO n DO BEGIN
3       IF i<n THEN j:=succ(i) ELSE j:=1;
3       IF x[j]<>x[i] THEN BEGIN
4                            m:=(y[j]-y[i])/(x[j]-x[i]);
4                            na:=y[i]-m*x[i];
4                            IF m=md THEN GOTO m1;
4                            xs:=(na-nb)/(md-m);
4                            ys:=m*xs+na END
3                       ELSE BEGIN xs:=x[i];ys:=md*x[i]+nb END;
3       IF ((abs(xs-x[i])<0.0001) AND (abs(ys-y[i])<0.0001)) OR
3        ((abs(xs-x[j])<0.0001) AND (abs(ys-y[j])<0.0001)) THEN
3        BEGIN yanf:=yanf-dichte-0.3; GOTO m2 END;
3       IF flag1 THEN BEGIN
4                   IF (xs>ymax) OR (ys>xmax) THEN GOTO m1 END
3                 ELSE IF (xs>xmax) OR (ys>ymax) THEN GOTO m1;
3       betragx:=abs(x[i]-x[j]);
3       betragy:=abs(y[i]-y[j]);
3       IF (abs(x[i]-xs)<=betragx) AND (abs(x[j]-xs)<=betragx)
3       AND (abs(y[i]-ys)<=betragy) AND (abs(y[j]-ys)<=betragy)
3        THEN putin(xs,ys);
3 m1: END;                          (* for i:=1 to n ... *)
2     flag:=TRUE;
2     drawline(clipp);
2     release(heapbottom);
2   END;                            (*  while yanf<=yend  *)
1 END;  (* F I L L A R E A *)
```

Bild 2 Grafikroutinenpaket

```
0 PROGRAM DEMO;
0 (* Dieses Programm demonstriert die Anwendung der Grafik-
0    routinen. Es zeichnet ein schraffiertes Rechteck.     *)
0
0 TYPE  vektor = ARRAY [1..4] OF REAL;
0 VAR   a,b : vektor;
0       i   : INTEGER;
0
0 (*$I TGRAFIK.INC  <---  in Bild 1 dargestelltes Routinen-
0                         paket als INCLUDE-Bestandteil    *)
0 BEGIN
1   a[1]:=150; a[2]:=450; a[3]:=450; a[4]:=150;
1   b[1]:=30;  b[2]:=30;  b[3]:=170; b[4]:=170;
1   clrpb;
1   line(round(a[4]),round(b[4]),round(a[1]),round(b[1]));
1   FOR i:=1 TO 3 DO
1   line(round(a[i]),round(b[i]),round(a[i+1]),round(b[i+1]));
1   fillarea(a,b,pi/4,4,3);
1   dump
1 END.
```

Bild 3 Demonstrationsprogramm zur Verdeutlichung der Arbeit mit den Grafikroutinen aus Bild 2

Zeitparalleler Druck unter UDOS

In Weiterentwicklung des in MP 5/87, S. 153, vorgestellten Druckertreibers SDOPT entstand ein Programmsystem, mit dem es möglich ist, Textdateien zeitparallel zu anderen Arbeiten am Rechner auszudrucken. Zusätzlich zu den im oben genannten Druckertreiber einstellbaren Parametern, wie Zeile je Seite, Seitennumerierung, Linefeed je Zeile, Format, Trennlinie zwischen den Seiten, Kopfsatz, Zeilenabstand, Zeichenabstand und Farbbandumschaltung, kann beim Start des Programms ein Rand eingestellt werden. Dadurch ist es möglich, bei Druckern mit geteilter Walze beide Papierbahnen zu nutzen. Das Programm belegt während des Druckens 2,5 KByte des Hauptspeichers. Die Nachnutzungsgebühr beträgt 100 M.

VEB Rohrkombinat Stahl- und Walzwerk Riesa, BfN oder Abt. WAA, Kolln. Haschlar, Dimitroffstr. 10, Riesa, 8400; Tel. 803196

Dr. Förster

Grafikeditor CZGREDIT

Manfred Berner, Dietmar Fürste
Forschungszentrum des Kombinates
VEB Carl Zeiss JENA

Der Beitrag informiert über den Grafikeditor CZGREDIT, der zum interaktiven Erstellen, Ändern und Speichern von grafischen Informationen dient. Rechentechnische Basis sind 8-Bit-Mikrorechnersysteme wie Bürocomputer A 5120 oder das Modulare System des Kombinates VEB Carl Zeiss JENA mit den Betriebssystemen UDOS bzw. CZ-SDOS, das Robotron-Rastersichtgerät (RSG) K8917 sowie ein von der Technischen Universität Dresden angebotenes GKS-Laufzeitsystem. Es werden Hinweise auf die zur Nutzung erforderliche Systemumgebung gegeben sowie Anwendungsmöglichkeiten und Grenzen diskutiert.

Zielstellung

Beim Aufbau grafischer Arbeitsplätze, z. B. für CAD/CAM-Aufgaben, spielen 8-Bit-Mikrorechner zum jetzigen Zeitpunkt und auch in den nächsten Jahren noch eine wesentliche Rolle. Als Grafik-Software dafür hat sich international das Grafische Kernsystem (GKS) durchgesetzt /1/. Für die Konfiguration BC A 5120 und RSG K8917 wird von der TU Dresden eine GKS-Version /2/ angeboten, auf deren Basis der Grafikeditor als FORTRAN-Anwenderprogramm entwickelt worden ist.
Die Zielstellung für CZGREDIT ist in dem Versuch zu sehen, erste Erfahrungen mit einem GKS-System auf 8-Bit-Rechnern zu sammeln, die Grenzen der Anwendung festzustellen und daraus sinnvolle Einsatzkriterien abzuleiten.

Ergebnis

Es liegt ein leistungsfähiges Programm vor, mit dem der Bediener im Dialogbetrieb grafische Informationen auf verschiedenen Fachgebieten erstellen, manipulieren und speichern kann. Entsprechend dem Grundgedanken des GKS werden die abzubildenden Objekte der Realwelt als Segmente des entstehenden Bildes definiert und damit die Voraussetzung für deren Manipulierbarkeit geschaffen /3/, /4/. Bild 1 zeigt das Hauptmenü mit den realisierten Funktionen:

```
GRAFIK-EDITOR    C Z G R E D I T
Die Funktionen:

 1:    SET SYMBOL
 2:    DRAW LINE(S)
 3:    SET TEXT
 4:    SET MARKER (1...10)
 5:    MOVE/DELETE SEGMENT
 6:    KREIS/BOGEN
 7:    OPEN METAFILE (READ/WRITE)
 8:    READ FROM MTF (OPEN BEFORE !)
 9:    CLOSE WORKSTATION(S)
10:    H E L P
11:    READ FROM CZPLOT-ASCII-BILDFILE
12:    CLEAR RSG

HARDCOPY (AUF 1157.269):  TASTE  R E C !

FUNKTION ?  :
```

Bild 1 Hauptmenü des CZGREDIT

∗ *SET SYMBOL*

Zur Auswahl stehen 7 feste Symbole, die im LOCATOR-MODE an die Stelle gezeichnet werden, die mit dem Fadenkreuz des RSG positioniert wird. Die Symbole repräsentieren Elemente eines Programmablaufplanes, der so direkt am Bildschirm erarbeitet, zwischengespeichert und anschließend ausgedruckt werden kann. Für den jeweiligen Spezialeinsatz können in nutzereigenen Subroutinen andere feste Symbole generiert werden. So lassen sich z. B. Schaltzeichen, Lötaugen-Raster für Leiterplatten oder Kartiersymbole definieren, vgl. Bild 2.

∗ *DRAW LINE*

Für das Zeichnen von Linienzügen (Polylinie) stehen die in Bild 3 abgebildeten 10 Linienarten zur Verfügung. Jedes Polygon wird als Segment abgespeichert.

∗ *SET TEXT*

Der Text wird im LOCATOR-Mode mit Hilfe des Fadenkreuzes positioniert. Es kann zwischen 10 Schriftarten gewählt werden (vgl. Bild 3). Eine nachträgliche Manipulation der Texte ist möglich.

∗ *SET MARKER*

Auf einen mit dem Fadenkreuz positionierten Punkt wird ein aus der in Bild 3 gezeigten Menge ausgewähltes Standardsymbol, ein sogenannter MARKER, gezeichnet.

∗ *MOVE/DELETE SEGMENT*

Dies ist die wichtigste Funktion des CZGREDIT, mit der für jedes Segment folgende Transformationen möglich sind:
– Translation
– Rotation
– Skalierung
– Löschen.

∗ *KREIS/BOGEN*

Kreise und Kreisbögen können durch Positionieren von 3 Punkten (wahlweise 3 Peripheriepunkte oder 2 Peripherie-Punkte mit Mittelpunkt) in 10 Linienarten generiert werden.

∗ *Metafiles*

Mit Hilfe der Funktion 7, 8 und 9 werden Metafiles eröffnet, gelesen, geschrieben und geschlossen, in denen Segmente und ihre Transformationen als grafische Records im Format des RSG /7/ abgespeichert sind. Darüber hinaus können speicherintensive Anwenderprogramme auch Metafiles direkt erzeugen, die sich später durch die CZGREDIT-Funktionen 7 und 8 in das zu erstellende Bild einfügen lassen, vgl. Bild 4.

∗ *READ FROM CZPLOT-ASCII-BILDFILE*

Dies ist eine Sonderfunktion, um Bildfiles, die im Format des Digitalzeichentisches DZT 90×120/RS vorliegen, für eine Weiterverarbeitung einzulesen /5/.

∗ *HELP*

Aufruf des Hauptmenüs, vgl. Bild 1.

∗ *CLEAR RSG*

Aufruf einer Kaltstartprozedur, bei der das RSG neu initialisiert sowie Bild- und Segmentspeicher gelöscht werden.

Anmerkung: Beim Anfertigen einer Hardcopy vom RSG K8917 auf dem Drucker 1157.269 wird wegen der gröberen Rasterung durch den Drucknadelabstand die Darstellung um den Faktor 1.276 vergrößert!

Nachnutzung des CZGREDIT

Vom VEB Carl Zeiss JENA wird zur Nachnutzung das Anwenderprogramm CZGREDIT als eine Sammlung von FORTRAN-Objektcode-Files und als lauffähig gebundene Pro-

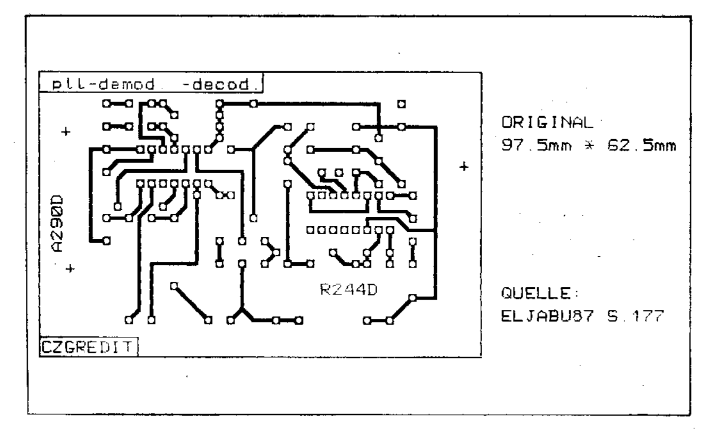


Bild 2 Leiterplatten-Urbild. Editiert mit variiertem Symbolgenerator, der Lötaugenraster für diverse Schaltkreise enthält

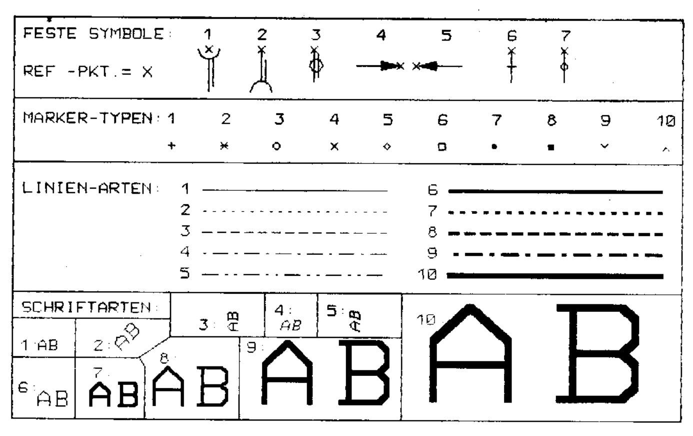


Bild 3 Repräsentation der Symbole, Marker, Linien- und Schriftarten

zedur angeboten, von der TU Dresden das Programm RSG-Anschluß an BC A5120 unter UDOS. Will der Anwender Teile des CZGREDIT ändern oder ergänzen, wird die FORTRAN-Version 4.0 benötigt /6/. CZGREDIT enthält je nach Rechnertyp unterschiedliche Schnittstellen-Adressen, die bei Nachnutzung mit dem VEB Carl Zeiss JENA abzustimmen sind. Die Programme werden zusammen mit der detaillierten Dokumentation auf nutzereigenen Disketten (Format UDOS oder CZ-SDOS, 8 oder 5,25 Zoll) bereitgestellt.

Grenzen und Ausblick

Die mit der Konfiguration RSG K8917 und CZGREDIT erreichbaren Funktionen müssen der niedrigsten Leistungsklassen einer GKS-Implementierung zugerechnet werden. So sind „höhere" GKS-Funktionen wie FILL AREA, SET COLOR und PIXEL ARRAY nicht enthalten. Die relativ kleine Sichtfläche des K8917 von etwa 210 mm × 133 mm mit einer Auflösung von nur 640×288 Bildpunkten /7/ schließt höhere Genauigkeitsansprüche aus, die z. B. für exakte Bau- und Konstruktionspläne erforderlich sind. Die Grenzen des 32K-Segmentspeichers des K8917 und auch die des Hauptrechner-Speichers sind schnell erreicht, wenn etwa für eine gut ausgenutzte Leiterplatte das vollständige Leiterbild editiert wird. Deshalb mußte im CZGREDIT auch auf Nullpunktverschiebung, Koordinatenumrechnungen und WINDOW-Techniken verzichtet werden.
Gegenüber der Nutzung des K8917 ausschließlich als Grafik-Ausgabegerät ist jedoch mit der interaktiven Bildschirmarbeit ein deutlicher Qualitätssprung spürbar, der für viele Anwendungen durchaus sinnvoll erscheint. So konnte mit dem RSG K8917 und dem CZGREDIT eine Arbeitsstation simuliert werden, deren praktischer Einsatz für z. B. folgende Arbeitsgebiete empfohlen werden kann:
Musterentwurf, Lagepläne, einfache Leiterplatten, Modellierungen in Architektur, Bergbau oder Formenbau, sogenannte Geschäftsgrafiken und Kurvendarstellungen, Schrift- bzw. Symbol-Generierung, Stromlauf- und Bestückungspläne usw.
Mit der Verfügbarkeit künftiger Grafik-Displays und anderer Grafikausgabegeräte mit entsprechender Intelligenz sollte jedoch die konsequente Durchsetzung der in der GKS-Normung festgelegten Grafiksprache an deren Schnittstellen erreicht werden. Es kann dem Nutzer solcher Geräte nicht zugemutet werden und ist auch volkswirtschaftlich nicht vertretbar, wenn für jede Konfiguration ein Spezial-Anpassungsprogramm geschrieben werden muß, dessen Schnittstelle dann zu keiner anderen Hard- oder Software paßt. Um zum Beispiel das im Robotron-Rastersichtgerät K8917 dargestellte Bild auf einem Robotron-Plotter K6418 zu zeichnen, muß erst ein Übersetzungsprogramm geschrieben werden, welches von keinem der beiden Hersteller-Betriebe angeboten wird.
Zusammenfassend soll aber noch einmal deutlich gemacht werden, daß mit der Entscheidung für das GKS ein wichtiger Schritt in die richtige Richtung getan wurde und daß man bei der nächsten Generation grafikfähiger Robotron-Geräte auf die Fortsetzung dieser Entwicklung setzen sollte /8/.

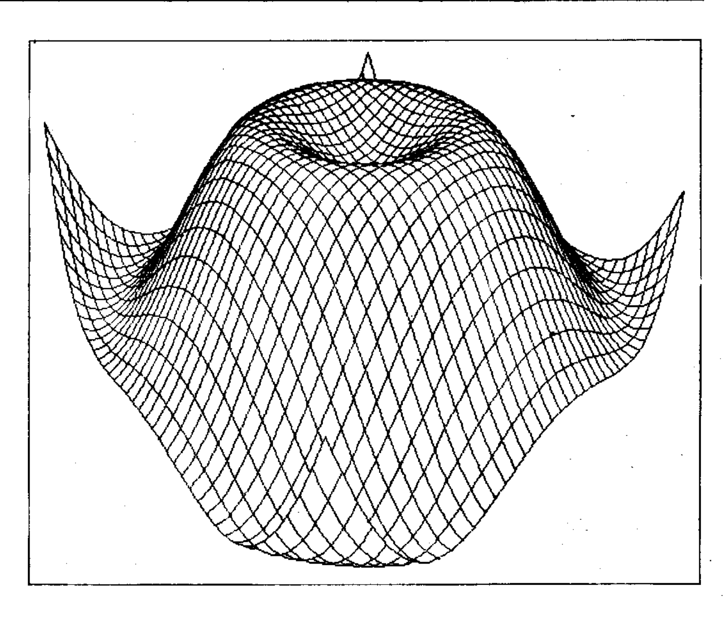


Bild 4
Darstellung und Funktion X * SIN (X), die als Metafile durch eine FORTRAN-Prozedur erzeugt wurde

Literatur

/1/ Entwurf DIN 66252: Graphisches Kernsystem (GKS) DK 681.3.04/.06:001.4, Oktober 1983
/2/ RSG – Steuerung unter UDOS. Anwenderdokumentation TU Dresden, Rechenzentrum, Bereich Kleinrechner
/3/ Straßer, W.: Graphische Datenverarbeitung heute, eine Bestandsaufnahme. In: Handbuch der Modernen Datenverarbeitung, Heft 127, Januar 1986, FORKEL-Verlag GmbH Wiesbaden
/4/ Encarnacao, J.: Graphische Datenverarbeitung – das Werkzeug für benutzerkonforme Anwendungen der Informatik. ebenda Heft 109, Januar 1983
/5/ Berner, M.; Fürste, D.: Programmsystem CZPLOT. Mikroprozessortechnik, Berlin 1 (1987) 7, S. 223
/6/ Anwenderdokumentation FORTRAN UDOS-1526. VEB Robotron Buchungsmaschinenwerk Karl-Marx-Stadt 1984
/7/ Betriebsdokumentation K 8917. VEB Robotron-Elektronik Zella-Mehlis
/8/ Mikut, M.; Werner, H.: Grafisches Kernsystem GKS 1600. Neue Technik im Büro, Berlin 28 (1984) 1

✉ KONTAKT ☎
Kombinat VEB Carl Zeiss JENA,
Forschungszentrum, Carl-Zeiss-Str. 1,
Jena, 6900; Tel. 83 31 56

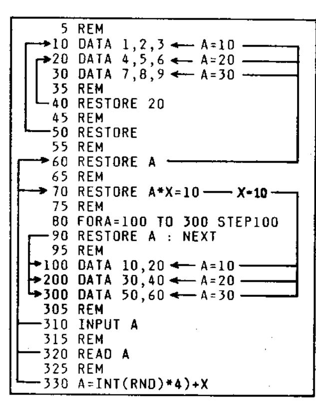
Erweiterte Zeilenbefehle für den KC-BASIC-Interpreter

Uwe Zierott, Lehnin

Da BASIC auf allen Kleincomputern, wenngleich in verschiedenen Dialekten, lauffähig ist, lassen sich viele Programme relativ einfach, meist unter Beachtung der vorhandenen Grafikbefehle, vom Entwicklungsrechner auf einen betriebssystemfremden Computer übertragen. In fast allen BASIC-Interpretern sind jedoch auch maschinenorientierte oder speicherplatzbezogene Befehle vereinbart, die eine Anpassung auf einen anderen Rechner erschweren oder unrationell machen können. Unmöglich wird es aber, wenn Interpreterroutinen und Monitorprogramme in das BASIC-Programm einbezogen werden. Andererseits lassen sich so der Befehlsvorrat erweitern, Laufzeiten und Speicherplatzbedarf stark verringern sowie das Verarbeiten von kompatiblen Dateien in BASIC- und Assemblerprogrammen u. v. m. realisieren. Vor- und Nachteile sollten also stets abgewogen werden.

Erweiterter RESTORE-A-Befehl

Der BASIC-Interpreter bietet zwei RESTORE – Varianten an. RESTORE ohne Argument nimmt die niedrigste Zeilennummer der vorhandenen DATA-Anweisungen im BASIC-Programm an. RESTORE 'Zeilennummer' organisiert die Wertzuweisung für die nächsten READ-Anweisungen, beginnend mit dem ersten Datenelement der DATA-Anweisung in der angegebenen BASIC-Zeile. Eine nützliche Erweiterung ist der RESTORE 'Ausdruck'-Befehl. Er gestattet u. a., die Unterprogrammtechnik noch konsequenter zu nutzen, ohne dabei die Stack-Tiefe zu erhöhen, den Wegfall von Suchschleifen, den Direktzugriff auf DATA-Zeilen durch Tastatureingabe oder Berechnungen im Programm. Eine Übersicht dazu bietet Bild 1.
Die Anweisung in Zeile 60 würde zu der Fehlermeldung 'SN ERROR IN 60' führen, weshalb die RESTORE-Routine des Interpreters direkt aufgerufen wird. Dazu muß im Doppelregister DE die Zeilennummer stehen. Die USR(X)-Funktion erfüllt diese Forderung. Bild 2 zeigt die Befehlsgruppen START, DATA, JUMP und CALL im Quellcode. Die je zwei Assemblerbefehle ab 0429H und 0439H dienen nur der Unterdrückung von 00H (durch DOKE) in der REM-Zeile, die im EDIT-Mode zur Programmverfälschung führen würden. Der HEX-Dump in Bild 3 kann in eine vergrößerte REM-Zeile ab 0406H eingegeben werden. Die im HEX-Dump abgebildete REM-Zeile erweitert das CAOS-MENU um



```
 5 REM
10 DATA 1,2,3 ← A=10
20 DATA 4,5,6 ← A=20
30 DATA 7,8,9 ← A=30
35 REM
40 RESTORE 20
45 REM
50 RESTORE
55 REM
60 RESTORE A
65 REM
70 RESTORE A*X=10 —— X*10
75 REM
80 FORA=100 TO 300 STEP100
90 RESTORE A : NEXT
95 REM
100 DATA 10,20 ← A=10
200 DATA 30,40 ← A=20
300 DATA 50,60 ← A=30
305 REM
310 INPUT A
315 REM
320 READ A
325 REM
330 A=INT(RND)*4)+X
```

Bild 1 Möglichkeiten für RESTORE

```
0406  7F 7F 5A 45 49 4C 45 4E
040E  42 45 46 45 48 4C 45 01
0416  01 EE C9 C5 C3 69 C6 CD
041E  6F C9 C3 E9 C8 D1 D1 D1
0426  11 FF FF 21 FF FF 22 27
042E  04 21 54 C8 E5 C3 0A CA
0436  11 FF FF 21 FF FF 22 37
043E  04 E1 E1 C1 0E 03 CD 27
0446  C3 E5 E5 2A 58 03 E3 3E
044E  8C F5 33 1B DC
```

Bild 3 HEX-DUMP für Zeile 0

```
        ORG     0406H           ;REM-Zeile 0
GSBTOK  EQU     08CH            ;TOKEN GOSUB
CULINO  EQU     0358H           ;AKT.ZEILE
TMEMO   EQU     0C327H          ;49BYTES-TEST
INITR   EQU     0C669H          ;INIT STACK
RUNMOD  EQU     0C854H          ;RUN-MODE
RESTO   EQU     0C8DFH          ;INIT DATA
PARAM   EQU     0C96FH          ;PE: USR(A)
RUN     EQU     0C9EBH          ;PROGRAMMSTART
GOTO    EQU     0CA07H          ;UP GOTO
=========================================
        DEFW    07F7FH          ;PROLOG
        DEFM    'ZEILENBEFEHLE'
        DEFB    01
START   LD      BC,RUN+3
        PUSH    BC              ;RUN IN DEN
        JP      INITR           ;STACK
=========================================
DATA    CALL    PARAM           ;ZEILE IN DE
        JP      RESTO+10
=========================================
JUMP    POP     DE              ;STACKTIEFE
        POP     DE              ;MANIPULIEREN
        POP     DE              ;(FOR-NEXT)
        LD      DE,0FFFFH       ;DOKEJP,A
        LD      HL,0FFFFH       ;00H VER-
        LD      (JUMP+4),HL     ;HINDERN
JUMPR   LD      HL,RUNMOD       ;RUN-MODE
        PUSH    HL              ;IN DEN STACK
        JP      GOTO+3
=========================================
CALL    LD      DE,0FFFFH       ;DOKEGS,A
        LD      HL,0FFFFH
        LD      (CALL+1),HL
        POP     HL
        POP     HL
        POP     BC
        LD      C,3             ;TEST
        CALL    TMEMO           ;MEMORY
        PUSH    HL
        PUSH    HL
        LD      HL,(CULINO)     ;AKT. ZEILE
        EX      (SP),HL         ;MERKEN
        LD      A,GSBTOK        ;08CH FÜR
        PUSH    AF              ;RETURN
        INC     SP
        JR      JUMPR           ;RUN-MODE
```

Bild 2 Quellcode für ZEILENBEFEHLE

```
0 !  MC-Code aus Bild 3
3 Z=10:DOKE772,1053:!**USR(X)-STARTADR.**
4 DEFFNA(I)=INT(I/100)*1000-1:!**für GOSUB**
5 JP=1063:!**Adr.für DOKE (GOTO A)**
6 GS=1079:!**Adr.für DOKE (GOSUB A)**
7 GO$="50 GOTO":GS$="DOKE GS,A":JP$="DOKE JP,I":P$=" ===> ":F$=" für "
8 Z$="     weiter in Zeile:":S$="40 GOSUB":D$=" DATA -Zeile in "
10 I=((INT(RND(1)*4))+Z)*Z
15 PRINTCOLORI/Z,6;"10 I=((INT(RND(1)*4))+Z)*Z / I=";I
20 FORJ=1TO3
30 R=USR(I+J):READ A$,A,V,H
35 PRINTCOLORV,H;"30 RESTORE I+J";P$;"R=USR(I+J)";F$;"I=";I;";J=";J
36 PRINTCOLORV,H;"    :READA$,A,V,H"
40 DOKEGS,A:CALL*436
50 X=0:DOKEJP,I:CALL*423
100 PAPER0:DOKEGS,FNA(I):CALL*436
101 DATA erste,222,0,2
102 DATA zweite,333,4,2
103 DATA dritte,444,6,2
110 PAPER2:DOKEGS,FNA(I):CALL*436
111 DATA vierte,222,0,4
112 DATA fünfte,333,2,4
113 DATA sechste,444,6,4
120 PAPER5:DOKEGS,FNA(I):CALL*436
121 DATA siebente,222,0,5
122 DATA achte,333,4,5
123 DATA neunte,444,7,5
130 PAPER6:DOKEGS,FNA(I):CALL*436
131 DATA zehnte,222,4,7
132 DATA elfte,333,0,4
133 DATA zwölfte und letzte,444,13,3
200 PAUSE:NEXT:GOTO10
222 INK18:PRINTA$;:INK2:GOTO555
333 INK20:PRINTA$;:INK4:GOTO555
444 INK23:PRINTA$;:INK7
555 PRINTD$;I+J:PRINTS$;A;P$;GS$;F$;"A=";A:RETURN
999 PRINTGO$;I;P$;JP$;F$;"I=";I:PRINTZ$;I+X:PAPER1
1000 PRINTCOLOR0,7;I+X;"GOSUB";FNA(I);P$;"DOKEGS,FNA(I)"SPC(Z);":CALL*436"
1001 X=X+Z:RETURN
```

Bild 4 BASIC-Programm BEISPIEL

Uwe Zierott (33) war bis 1977 im VEB DVZ Potsdam als EDV-Facharbeiter tätig. Gegenwärtig leitet er die AG Kleincomputer an der Karl-Marx-Oberschule Lehnin und unterrichtet als nebenberuflicher Dozent für Computeranwendung und Programmierung an der Kreisvolkshochschule.

% ZEILENBEFEHLE. Das dahinter abgelegte BASIC-Programm kann von MENU aus gestartet werden, wenn eine BASIC-Zeile 0 vorhanden ist (Selbststart bei Ansprung von 0416H nach dem Ladebefehl). Die neue Funktion wird mit

```
 0 !               :  REM für MC
 3 DOKE 772,1053   :  REM Start USR(A)
30 R=USR(I+J)      :  REM RESTORE A
```

aus dem BASIC-Programm aufgerufen.

RESTORE aus Arbeitszellen

Eine weitere Möglichkeit der Arbeit mit DATA-READ-RESTORE besteht in der Abfrage der entsprechenden Arbeitszellen des Interpreters.

So finden wir in 03CAH und 03CBH die Zeilennummer des zuletzt gelesenen DATA-Wertes, die z. B. als Abbruchbedingung dienen kann. Noch interessanter sind die Arbeitszellen 03DDH und 03DEH. Hier steht der Zeiger auf dem aktuellen DATA-Wert. Diese Adressen können mit

```
10 R=DEEK(989)   :  REM ZEIGER und
20 Z=DEEK(970)   :  REM ZEILE gelesen,
                    sowie mit
30 DOKE 989,R    :  REM ZEIGER und
40 I=USR(Z)      :  REM ZEILE
```

für den Programmablauf genutzt werden.

Erweiterter GOTO-A-Befehl

Die Anwendung einer GOTO-'Ausdruck'-Anweisung kann analog zu RESTORE erfolgen. Im Beispiel wird durch

```
50 DOKE JP,I  :  CALL*423
```

das Doppelregister DE beim Unterprogrammaufruf JUMP mit der Zeilennummer geladen. Die Implementierung von ON GOTO A bzw. ON GOBUB A ist nicht sinnvoll, da die vorgestellten Routinen in ihrer Kombination allen Anforderungen genügen.

Soll die Anweisung

```
IFX>Y THEN 100  :  ELSE 200
```

ersetzt werden, so können die Schlüsselworte THEN und ELSE nicht entfallen, da deren Routinen anders generiert sind. Für Y=200 gilt daher

```
IFX>Y THEN A=Y/2  :  ELSE A=Y
DOKE JP,A  :  CALL*423
```

wobei die gleiche Programmverzweigung durch veränderte X- und Y-Werte universell nutzbar wird. Zu beachten ist, daß A als Zeilennummer vorhanden ist.

Erweiterter GOSUB-A-Befehl

Mit dieser Funktion wird der Nutzer in die Lage versetzt, auch die Auswahl der Unterprogramme variabel zu gestalten. Im Programm *Beispiel* erfolgt der Aufruf durch

```
 40 DOKE GS,A       :  CALL*436 bzw.
100 DOKE GS,FNA(I)  :  CALL*436
```

nachdem GS=1079, DEFFNA(I) und READ A vorausgingen.

Das Einschalten des IRM ist in den Routinen nicht berücksichtigt, da im Normalfall keine Notwendigkeit vorliegt. Nur der MEMORYTEST, der bei sicheren Programmen nicht notwendig ist, kann bei unzulässiger Stacktiefe in seiner Fehlerausschrift behindert werden. Zur Vollständigkeit hier die zeitgünstigste Einschaltmethode für den IRM:

```
IN   A,088H
SET  2,A
OUT  088H,A
```

Die GOSUB A- und GOTO A-Anweisung lassen sich bequemer und auch zeitsparender mit der USR(X)-Funktion starten. Das Beispiel sollte jedoch auch zeigen, wie auf einfache Weise Registerinhalte aufgerufener Maschinenprogramme durch BASIC manipuliert, also der Ablauf geändert werden kann.

Schlußbemerkung

Die Anwendung der vorgestellten Befehlserweiterungen, ob einzeln oder kombiniert (auch mit BASIC-Standard), eröffnet nicht nur neue Möglichkeiten der BASIC-Programmierung, sondern erzwingt auch klare Programmstrukturen. Gute Lesbarkeit und logischer Aufbau werden nicht Ziel, sondern Bedingung der Programmentwicklung.

Trotzdem sollten die Erweiterungen sparsam verwendet werden und ganz unterbleiben, wo das vertraute BASIC die gleichen Dienste leistet. Ansonsten entstehen längere Programme, die kaum überschaubar und noch weniger editierbar sind. Die Laufzeit des BASIC-Programms wird nicht durch den Befehlsaufruf, sondern durch das Entfallen mehrerer BASIC-Anweisungen wie Programmverzweigungen, Schleifen und radikale Verkürzung bei Vielfachnutzung von Programmteilen erreicht.

Die angeführten Beispiele haben in einer konkreten Anwendung (Tabellenberechnungen und Menütechnik) aus dem Bereich des Versicherungswesens zu Zeit- und Speicherplatzeinsparung von mehr als 40 Prozent geführt, sind jedoch nur erste Versuche der Nutzung von Interpreterroutinen.

Literatur

/1/ Völz, H.: Universelle Nutzung des BASIC-Interpreters. Mikroprozessortechnik, Berlin 1 (1987) 6, S. 182–184 und 7, S. 221–222

✉ KONTAKT ☎

Kreisvolkshochschule Brandenburg, Neustadt-Markt 7, Brandenburg, 1800, Koll. Zierott

PASCAL

(Teil 2)

Dr. Claus Kofer
Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden

5.2.3. Konstantendeklaration

Die Konstantendeklaration ermöglicht die Einführung von Bezeichnern für solche Datenobjekte, deren Wert von vornherein feststeht und sich auch während der Programmabarbeitung nicht ändert. Die Bezeichner können dann synonym für die entsprechende Konstante verwendet werden.



Bild 5.4 Syntaxdiagramm „konstantendeklaration"

Das Bild 5.4 zeigt die Syntax der Konstantendeklaration. Sie wird durch den reservierten Bezeichner CONST eingeleitet. Mit

CONST Epsilon = 0.01;

wird z. B. die Konstante Epsilon deklariert, die vom Typ REAL ist und den Wert 0.01 hat.
Es können auch mehrere Konstanten deklariert werden. Durch

```
CONST Max = 100;
      Delta = -1.5;
      Kontrolldruck = FALSE;
      Prompt = '>';
      Text = 'Programm V1.0;
```

werden die Konstanten

- Max vom Typ INTEGER mit dem Wert 100,
- Delta von Typ REAL mit dem Wert −1.5,
- Kontrolldruck vom Typ BOOLEAN mit dem Wert FALSE,
- Prompt vom Typ CHAR mit dem Wert '>' und
- Text vom Typ Zeichenkette mit dem Wert 'Programm V1.0'

eingeführt.

In Standard-PASCAL können nur INTEGER-, REAL-, BOOLEAN-, CHAR- und Zeichenkettenkonstanten deklariert werden.
Als Erweiterung gegenüber dem Standard gestattet TURBO-PASCAL die Deklaration getypter Konstanten. Der Typ ist mit Ausnahme von File beliebig, also z. B. auch strukturiert. Siehe dazu auch Punkt 11.2.

5.2.4. Typendeklaration

Mit der Typendeklaration werden Bezeichner für Datentypen eingeführt.

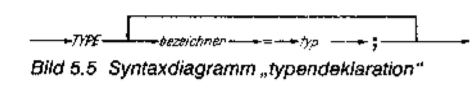


Bild 5.5 Syntaxdiagramm „typendeklaration"

Das Bild 5.5 zeigt die Syntax. Durch

TYPE Farbe = (ROT, BLAU, GRUEN);

wird z. B. der Typenbezeichner Farbe eingeführt. Der Bezeichner Farbe repräsentiert einen Aufzählungstyp und kann im nachfolgenden Programmtext ebenso verwendet werden wie der Bezeichner der Standardtypen INTEGER, REAL usw.

Weitere Beispiele für Typendeklarationen sind:

- TYPE Wochentag = (Mo,Di,Mi,Don,Fr, Sa,So);
- VglTyp = (kleiner,gleich,größer);
- EAFunktion = (Lesen, Schreiben, Positionieren);
- Sektorbereich = 1..12;
- Buchstabe = 'A'..'Z';
- Ziffer = '0'..'9'.

5.2.5. Variablendeklaration

Durch die Variablendeklaration werden Bezeichner für Variablen eingeführt. Gleichzeitig wird für jede Variable ihr Datentyp angegeben. Damit steht fest, welche Werte sie im Verlauf der Programmabarbeitung annehmen kann.

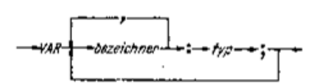


Bild 5.6 Syntaxdiagramm „variablendeklaration"

Das Bild 5.6 zeigt die Syntax. Die Variablendeklaration wird eingeleitet durch den reservierten Bezeichner VAR. Durch

```
VAR I  :  INTEGER;
    X  :  REAL;
    C  :  CHAR;
    B  :  BOOLEAN;
```

werden die Variablen I, X, C und B deklariert. Gleichzeitig wird aber auch festgelegt, daß I nur Werte aus dem Wertebereich von INTEGER annehmen kann, X aus dem Wertebereich von REAL usw.
Unter der Benutzung der weiter oben angeführten Aufzählung- und Teilbereichstypen ist folgende Variablendeklaration denkbar:

```
VAR Flagge  :  Farbe;
    Tag     :  Wochentag;
    Sekt    :  Sektorbereich;
```

Die variable Flagge ist vom Aufzählungstyp Farbe. Sie kann während der Programmabarbeitung nur einen der Werte ROT, BLAU oder GRUEN annehmen. Analog kann die Variable Tag nur einen der Werte Mo, Di, Mi, Don, Fr, Sa oder So annehmen. Der Typ der Variablen Sekt ist der Teilbereich 1...12 des Standardtyps INTEGER. Demzufolge sind ihre möglichen Werte eingeschränkt auf 1 ⇐ Sekt ⇐ 12.
Mehrere Variablen gleichen Typs können abkürzend in folgender Form deklariert werden:

```
VAR I,J,K      :  INTEGER;
    X,Z,U,V,W  :  REAL;
```

In PASCAL ist es nicht möglich, den Variablen bei ihrer Deklaration einen Anfangswert zu geben.

5.3. *Anweisungsteil*

5.3.1. Überblick

Den Aufbau des Anweisungsteils zeigt Bild 5.7. Eine Liste von Anweisungen wird durch BEGIN und END geklammert. Das Semikolon ist das Trennzeichen zwischen den Anweisungen. Die einzelnen Anweisungen werden in Bild 5.8 gezeigt.

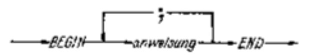


Bild 5.7 Syntaxdiagramm „anweisungsteil"

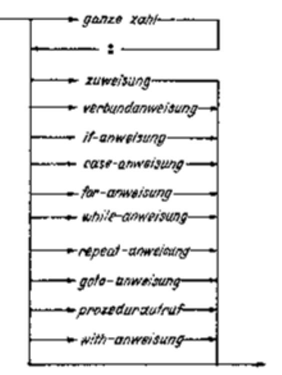


Bild 5.8 Syntaxdiagramm „anweisung"

Für den Anweisungsteil hat die syntaktische Einheit „ausdruck" eine gewisse zentrale Bedeutung. Sie wird im nächsten Unterpunkt besprochen. Daran schließen sich die einzelnen Anweisungen an. Prozeduraufruf und With-Anweisung werden jedoch aufgeschoben bis zu Prozeduren und Funktionen im Punkt 5.5. bzw. bis zum Datentyp Rekord im Punkt 8.

5.3.2. Ausdrücke

Die PASCAL-Notation eines Ausdruckes orientiert sich an der üblichen mathematischen Schreibweise. Seine Deklaration erfolgt mit Hilfe der Syntaxdiagramme „einfacher ausdruck", „term" und „faktor". Sie werden in den Bildern 5.9 bis 5.12 gezeigt.

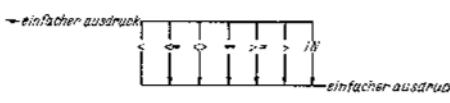


Bild 5.9 Syntaxdiagramm „ausdruck"

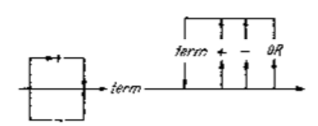


Bild 5.10 Syntaxdiagramm „einfacher ausdruck"

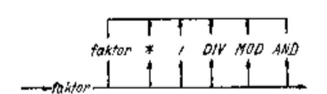


Bild 5.11 Syntaxdiagramm „term"

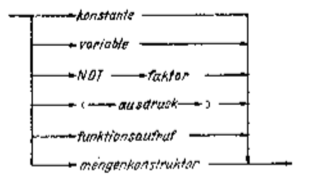


Bild 5.12 Syntaxdiagramm „faktor"

Faktor enthält Alternativen, die eine Bezugnahme auf die Datenobjekte des PASCAL-Programms gestatten.
Durch die hierarchische Deklaration von Ausdruck erhalten die Operatoren unterschiedliche Prioritäten. Die nachfolgende Aufstellung zeigt sie nach fallender Priorität angeordnet:

```
( )  NOT
*  /  DIV  MOD  AND
+  -  OR
<  <=  <>  =  >=  >  IN
```

Operatoren einer Zeile haben die gleiche Priorität.
Der Compiler weist Ausdrücke zurück, in denen auf Datenobjekte Operatoren angewendet werden, die für diese nicht erklärt sind. Dabei dürfen auf Teilbereichstypen grundsätzlich dieselben Operatoren angewendet werden, wie auf den entsprechenden Basistyp.
Als einzige Ausnahme organisiert der Compiler eine ggf. erforderliche Umwandlung von INTEGER in REAL.
Beispiele für korrekte Ausdrücke sind:

```
I+1
0.5*X+K
'Eine Zeichenkette'
I<100
Error OR PRINT AND LIST
(I<0) OR (100<I)
```

Im letzten Ausdruck sind die Klammern wegen der höheren Priorität von OR notwendig.
Nicht korrekte Ausdrücke sind:

```
'A' + 1
I AND 255
```

5.3.3. Zuweisung

Eine Zuweisung gibt einer Variablen einen neuen Wert. Das Bild 5.14 zeigt die Syntax. Die Zeichenkombination := ist der Zuweisungsoperator.

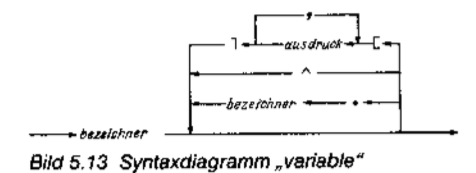
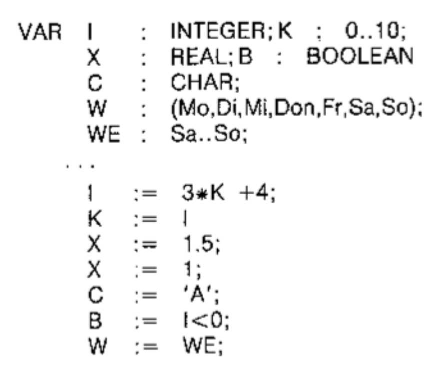


Bild 5.13 Syntaxdiagramm „variable"

variable → := → ausdruck

Bild 5.14 Syntaxdiagramm „zuweisung"

Der Compiler lehnt Zuweisungen ab, bei denen der Typ des Ausdruckes nicht mit dem der Variablen verträglich ist. Typen sind miteinander verträglich, wenn sie durch den gleichen Typenbezeichner deklariert wurden. Teilbereichstypen sind mit ihrem Basistyp verträglich. Als einzige Ausnahme organisiert der Compiler die Konvertierung von INTEGER in REAL.
Das nachfolgende Programmstück zeigt korrekte Zuweisungen:



```
VAR  I   :  INTEGER; K  :  0..10;
     X   :  REAL; B  :  BOOLEAN
     C   :  CHAR;
     W   :  (Mo,Di,Mi,Don,Fr,Sa,So);
     WE  :  Sa..So;
     ...
     I   :=  3*K +4;
     K   :=  I
     X   :=  1.5;
     X   :=  1;
     C   :=  'A';
     B   :=  I<0;
     W   :=  WE;
```

5.3.4. Verbundanweisung

Die Verbundanweisung hat ausschließlich eine syntaktische Funktion. Sie läßt eine Liste von Anweisungen wie eine einzige Anweisung wirken.

Bild 5.15 Syntaxdiagramm „verbundanweisung"

Die Syntax ist in Bild 5.15 dargestellt. Ein Beispiel ist:

```
BEGIN I:=1; VOLL:=TRUE; X:=0.5 END
```

5.3.5. If-Anweisung

Die If-Anweisung bewirkt eine Verzweigung im Programmablauf. Das Bild 5.16 zeigt die Syntax. Die auf die reservierten Bezeichner THEN bzw. ELSE folgenden Anweisungen heißen THEN- bzw. ELSE-Zweig.

Bild 5.16 Syntaxdiagramm „if-anweisung"

Bei der Abarbeitung der If-Anweisung wird zuerst der Ausdruck ausgewertet. Er muß einen Wert vom Typ BOOLEAN liefern. Ist er TRUE, wird der THEN-Zweig durchlaufen, sonst der ELSE-Zweig.

Beispiele:

```
IF X<>0 THEN Y:= 1/X  ELSE Y:=1.E38
IF A>B  THEN MAX:=A  ELSE MAX:=B
```

Durch Nutzung der Verbundanweisung können THEN- und ELSE-Zweig durch weitere Anweisungen aufgefüllt werden:

```
IF I > MAX  THEN BEGIN
   VOLL  :=  TRUE;  I:=0
END  ELSE  BEGIN
   K:= I DIV 4;  I:=I+1;  X:=1.0
END
```

Die Syntax läßt es zu, daß der gesamte ELSE-Zweig fehlen darf:

```
IF X < 0   THEN X:= -X
```

Es ist zulässig, If-Anweisungen beliebig ineinander zu verschachteln. Entstehende Mehrdeutigkeiten werden durch die Festlegung beseitigt, daß ein ELSE-Zweig stets zur letzten IF-Anweisung gehört:

```
IF a1 THEN
   IF a2 THEN anw2
      ELSE anw3
```

entspricht

```
IF a1 THEN BEGIN
   IF a2 THEN anw2 ELSE anw3
END
```

Soll der ELSE-Zweig zur ersten IF-Anweisung gehören, ist zu schreiben

```
IF a1 THEN BEGIN
   IF a2 THEN anw2 END
ELSE anw3
```

5.3.6. Case-Anweisung

Die Case-Anweisung ist eine ***Mehrwegverzweigung***. Ihre Struktur wird im Bild 5.17 gezeigt. Sie besteht aus einem Ausdruck und einer Liste, die durch Konstanten und Anweisungen gebildet wird. Die Konstanten und der Ausdruck müssen den gleichen einfachen Typ haben. REAL ist jedoch nicht zugelassen.

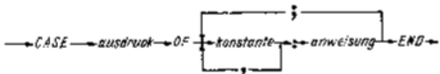


Bild 5.17 Syntaxdiagramm „case-anweisung"

Die Abarbeitung der Case-Anweisung beginnt mit der Berechnung des Ausdruckes. Dann wird die Anweisung abgearbeitet, deren vorangestellte Konstante der Wert des Ausdruckes ist, und danach die Case-Anweisung verlassen. Gibt es keine solche Anweisung, wird die Programmabarbeitung bei der nachfolgenden Anweisung fortgesetzt.
Es folgt ein Beispiel mit dem Aufzählungstyp Wochentag:

```
TYPE   Wochentag=(Mo,Di,
               Mi,Don,Fr,Sa,So)
VAR    Tag  :  Wochentag;
.
.
.
CASE Tag OF
  Mo  :  Anweisung 1;
  Fr  :  Anweisung 2;
  Sa  :  Anweisung 3;
END
```

Falls die Variable Tag den Wert Mo hat, wird Anweisung 1 abgearbeitet und danach die Case-Anweisung verlassen; falls Tag den Wert Fr hat, wird Anweisung 2 abgearbeitet und danach die Case-Anweisung verlassen usw. Hat die Variable Tag keinen der Werte Mo, Fr oder Sa, wird die Case-Anweisung sofort verlassen.
Ist für mehrere Konstanten die erforderliche Aktion gleich, können diese Konstanten zu einer Liste zusammengefaßt werden. Das nachfolgende Programmstück demonstriert das am Beispiel der Klassifikation von Operatoren.

```
TYPE OpTyp = (AddOp, Mulop);
VAR Operator : Op Typ;
        C : CHAR;
.
.
.
CASE C OF
'+','-' : Operator := AddOp;
'*','/' : Operator := MulOp
END
```

Falls die Variable C den Wert '+' oder '-'hat, wird der Variablen Operator der Wert AddOp zugewiesen. Ist der Wert von C '*'oder '/', dann erhält Operator den Wert MulOp. Es wäre sicher noch wünschenswert, auch die Operatoren OR, DIV, MOD und AND zu klassifizieren. Das ist mit diesem einfachen Programmstück aber nicht möglich. Als Konstanten in der Case-Anweisung dürfen keine Zeichenketten stehen.
TURBO-PASCAL läßt bei der Case-Anweisung einen ELSE-Zweig zu. Er wird durchlaufen, falls der Wert des berechneten Ausdrukkes mit keiner Konstanten übereinstimmt. Zur Demonstration wird das vorangegangene Beispiel erweitert.

```
TYPE OpTyp = (AddOp, MulOp, NoOp);
VAR  Operator : OpTyp;
        C : CHAR;
     .....
CASE  C  OF
      '+','-' : Operator := AddOp;
      '*','/' : Operator := MulOp
      ELSE    Operator := NoOP
END
```

Die Variable Operator wird mit dem Wert NoOP belegt, falls C keinen der Werte '+', '-', '*' oder '/' hat.
Die Syntax der Case-Anweisung läßt vor ELSE und END kein Semikolon zu.
TURBO-PASCAL bietet eine weitere Vereinfachung: Die einer Anweisung vorangehende Liste von Konstanten darf auch eine Teilbereichsangabe sein. Damit darf anstelle von

```
CASE c OF
'A','B','C','D','E','F','G':  anweisung;
  ...
```

einfacher geschrieben werden

```
CASE c OF
'A'..'G':  anweisung;
```

5.3.7. For-Anweisung

Die For-Anweisung stellt eine *Schleifenkonstruktion* dar. Bild 5.18 zeigt die Syntax. Die Variable muß vom einfachen Typ sein. REAL ist jedoch nicht erlaubt.
Die Ausdrücke dienen zur Angabe von Anfangs- und Endwerten, zwischen denen die Variable in ansteigender (TO) bzw. abfallender (DOWNTO) Folge Werte annimmt. Eine Wahl der Schrittweite ist nicht möglich.

Die Ausdrücke werden einmal zu Beginn berechnet. Die Schleifenbedingung wird vor jeder Abarbeitung der Schleifenanweisung getestet.
Beispiele für For-Anweisungen sind

```
FOR I:= 1 TO Max DO Fak:=Fak*I
FOR I:= Max DOWNTO 1 DO S:=S+1/I
FOR C:= 'A' TO 'Z' DO ...;
FOR C:= '9' DOWNTO '0' DO ...;
FOR Tag:= Mo TO Sa DO ...;
FOR B:= FALSE TO TRUE DO ...;
```

Sollen bei einem Schleifendurchlauf mehrere Anweisungen abgearbeitet werden, sind sie zu einer Verbundanweisung zu klammern:

```
FOR  I:=1  TO  MAX  DO  BEGIN
    S:=  S+Q;  Q:=  Q*Q/I
END
```

Bild 5.18 Syntaxdiagramm „for-anweisung"

5.3.8. While-Anweisung

Die While-Anweisung stellt ebenfalls eine *Schleifenkonstruktion* dar. Die Syntax wird in Bild 5.19 gezeigt. Der Ausdruck muß einen Wert vom Typ BOOLEAN liefern.

Bild 5.19 Syntaxdiagramm „while-anweisung"

Bei der While-Anweisung wird vor jedem Schleifendurchlauf der Ausdruck ausgewertet. Liefert er den Wert FALSE, wird die Schleife verlassen.
Das nachfolgende Programmstück zeigt die Anwendung der While-Anweisung zur Berechnung der Summe S=1/N+...+1/2+1.

```
S:=0;  I:=N;
WHILE I>0 DO BEGIN
    S:=S+1/I;  I:=I-1
END
```

5.3.9. Repeat-Anweisung

Die Repeat-Anweisung stellt eine weitere Schleifenkonstruktion dar (Syntax siehe Bild 5.20).

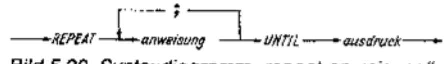


Bild 5.20 Syntaxdiagramm „repeat-anweisung"

Eine Liste von Anweisungen wird durch die reservierten Bezeichner REPEAT und UNTIL geklammert.
Bei jedem Schleifendurchlauf wird zuerst die Anweisungsliste abgearbeitet und danach der Ausdruck berechnet. Der Ausdruck muß einen Wert von Typ BOOLEAN liefern. Ist sein Wert TRUE, wird die Repeat-Anweisung verlassen.
Als Beispiel für die Anwendung der Repeat-Anweisung wird wieder die Berechnung der Summe S=1/N+...+1/2+1 gezeigt:

```
S:=0;  I:=N
REPEAT
    S:=S+1/I;  I:=I-1
UNTIL  I=0
```

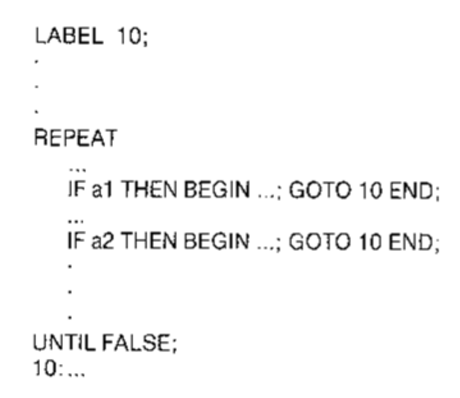
Da die Anweisungsliste wenigstens einmal abgearbeitet wird, führt das Programm für N=0 zu einem Fehler.

5.3.10. Goto-Anweisung

Das Bild 5.21 zeigt die Syntax der Goto-Anweisung. Sie bewirkt eine unbedingte Verzweigung zu der Anweisung mit der angegebenen Marke.

Bild 5.21 Syntaxdiagramm „goto-anweisung"

Das nachfolgende Programmbeispiel demonstriert, daß die PASCAL-Anweisungen zur Steuerung des Programmablaufs durchaus sinnvoll durch die Goto-Anweisung ergänzt werden können. Es wird eine Schleife mit verschiedenen Ausgängen und differenzierter Nachbehandlung gezeigt:



```
LABEL  10;
.
.
.
REPEAT
    ...
    IF a1 THEN BEGIN ...; GOTO 10 END;
    ...
    IF a2 THEN BEGIN ...; GOTO 10 END;
    .
    .
    .
UNTIL FALSE;
10:...
```

Bei TURBO-PASCAL dürfen Sprünge nicht aus einem Block herausführen.

5.4. Hauptprogramm

Ein PASCAL-Hauptprogramm besteht aus einem *Programmkopf* und dem Programmbaustein Block. Bild 5.22 zeigt dies. Die Syntax des Programmkopfes zeigt Bild 5.23. Er wird durch den reservierten Bezeichner PROGRAM eingeleitet. Es folgt der Name des Programms. Er kann für eine inhaltliche Kennzeichnung des Programms verwendet werden, hat darüber hinaus aber keine Bedeutung.

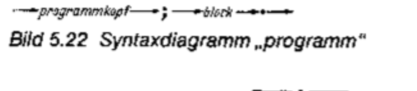


Bild 5.22 Syntaxdiagramm „programm"

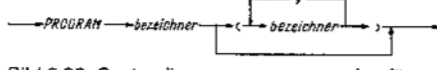


Bild 5.23 Syntaxdiagramm „programmkopf"

Dem Programmnamen kann eine in runde Klammern eingeschlossene Liste von Bezeichnern folgen. Sie stellen die Verbindung des PASCAL-Programms zum sogenannten Environment dar. In der Regel sind das Dateien, mit denen das Programm arbeiten kann, ohne sie eröffnen oder anlegen zu

müssen. In TURBO-PASCAL hat diese Liste keine Bedeutung. Abgeschlossen wird ein PASCAL-Programm durch einen Punkt.
Der Programmkopf des Beispiels aus Punkt 2.7 ist demnach

```
PROGRAM Tabelle (INPUT,OUTPUT)
```

Das Programm heißt Tabelle. Das benutzte Environment sind die Dateien INPUT und OUTPUT.
Es folgt ein Block, bei dem der Deklarationsteil nur die Variablendeklaration

```
VAR x,y,max:INTEGER;
```

enthält. Der folgende Anweisungsteil

```
BEGIN
  Readln(INPUT,max);
  FOR x:=0 TO max DO BEGIN
    y:=x*x; Writeln(OUTPUT,
    'x:=', x,'y:=', y)
  END
END
```

besteht aus einer Liste von zwei Anweisungen, einem noch zu besprechenden Prozeduraufruf und einer For-Anweisung. Das Schleifeninnere ist eine Verbundanweisung. Der Punkt beendet das Programm und damit einen Satz in der Programmiersprache PASCAL.

5.5. *Prozeduren und Funktionen*

5.5.1. Einführung

Prozeduren und Funktionen gestatten die Untergliederung eines Programms in relativ selbständige Teile. Das ursprüngliche Motiv war die Einsparung von Programmkode. Gleiche Programmpassagen werden durch Aufrufe einer Prozedur ersetzt, welche die erforderlichen Anweisungen enthält.
Zunehmend trat aber ein anderer Gesichtspunkt in den Vordergrund: Prozeduren und Funktionen gestatten es, inhaltlich zusammengehörende und eine gewisse abgeschlossene Leistung erbringende Programmteile im Programmtext als solche sichtbar zu machen. Damit läßt sich die Übersichtlichkeit eines Programms wesentlich verbessern.
In den nachfolgenden Unterpunkten wird zunächst die Syntax von Deklaration und Aufruf gezeigt. Danach werden Gültigkeitsbereiche von Bezeichnern und Speicherplatzzuordnung für lokale Variablen besprochen. Den Abschluß bilden Standardprozeduren und -funktionen.

3.5.2. Deklaration und Anruf

Ähnlich wie das Hauptprogramm werden Prozeduren und Funktionen durch den Kopf und den Programmbaustein Block gebildet. Die Bilder 5.24 und 5.26 zeigen das. Die an den reservierten Bezeichner FORWARD geknüpfte Alternative wird später behandelt.

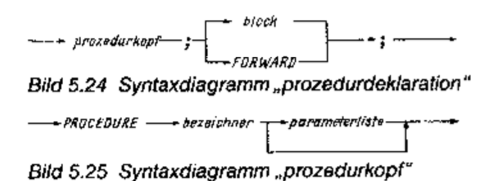


Bild 5.24 Syntaxdiagramm „prozedurdeklaration"

Bild 5.25 Syntaxdiagramm „prozedurkopf"



Bild 5.26 Syntaxdiagramm „funktionsdeklaration"

Bei Prozeduren wird der Kopf durch den reservierten Bezeichner PROCEDURE eingeleitet. Ihm folgt der Name der Prozedur. Wahlweise kann eine Parameterliste angegeben werden. Die Syntaxdiagramme 5.24 und 5.25 zeigen dies.
Bei Funktionen trägt der Funktionsname selbst einen Wert. Sein Typ wird, wie das Bild 5.27 zeigt, der Parameterliste nachgestellt. Erlaubt sind nur einfache Typen.

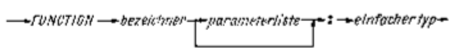


Bild 5.27 Syntaxdiagramm „funktionskopf"

Eine genaue Betrachtung der Syntax von Block zeigt, daß der Deklarationsteil weitere Prozeduren und Funktionen enthalten kann. Die Diskussion aller damit zusammenhängenden Probleme wird jedoch bis zum nächsten Unterpunkt aufgeschoben.
Eine Prozedur kann über sogenannte globale Variablen und Parameter mit ihrer Umwelt kommunizieren. Die Parameter werden in einer Parameterliste angegeben. Die einzelnen Elemente dieser Liste heißen bei der Deklaration formale Parameter, beim Aufruf aktuelle Parameter.

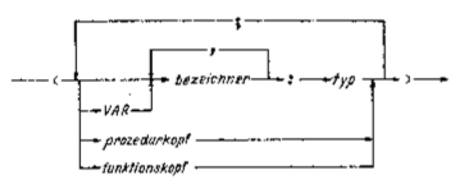


Bild 5.28 Syntaxdiagramm „parameterliste"

Das Bild 5.28 zeigt die Syntax der Parameterliste. Die einzelnen Alternativen repräsentieren die vier möglichen Parametertypen:

- Werteparameter
- Referenzparameter
- Prozedurparameter und
- Funktionsparameter.

Werte- und Referenzparameter übermitteln Datenobjekte. Ihnen wird deshalb eine Typangabe nachgestellt.
Während Werteparameter nur die Übertragung von Informationen an die Prozedur oder Funktionen gestatten, wirken Referenzparameter in beiden Richtungen.
Die Aktivierung von Prozeduren und Funktionen zeigen die Syntaxdiagramme der Bilder 5.29 und 5.30. Beide Formen sind identisch. Während aber der Prozeduraufruf eine selbständige Anweisung ist, können Funktionen nur als Faktor in einen Ausdruck aufgerufen werden.

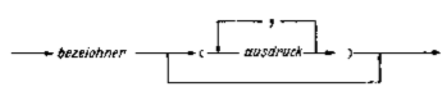


Bild 5.29 Syntaxdiagramm „prozeduraufruf"

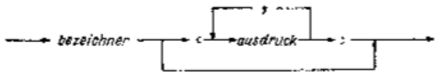


Bild 5.30 Syntaxdiagramm „funktionsaufruf"

Der Compiler löst die Aufrufe so auf, daß zuerst die Ausdrücke ausgewertet werden und danach in die entsprechende Prozedur oder Funktion verzweigt wird.
Vom Compiler wird überprüft, ob

- Anzahl und Typ der aktuellen Parameter mit den formalen übereinstimmen
- ein aktueller Referenzparameter eine Variable ist und eine
- Übereinstimmung zwischen den Parameterlisten der aktuellen und formalen Prozedur- und Funktionsparameter herrscht.

Es folgen Beispiele:
Die Prozedur Max 2 soll von zwei Parametern den Wert des größeren ermitteln und zurückgeben.

```
PROCEDURE Max2
        (A,B:REAL;VAR Max:REAL);
BEGIN
   IF A>B THEN Max:=A ELSE-
 Max:=B
END.
```

Im Prozedurkopf werden Name und Parameterliste angegeben. Die Parameterliste enthält die Werteparameter A und B vom Typ REAL und den Referenzparameter Max, der ebenfalls vom Typ REAL ist. Der Deklarationsteil ist leer. Es werden zur Ermittlung des Resultates keine Marken, Konstanten, Typen und Variablen benötigt. Der Anweisungsteil besteht aus einer If-Anweisung. Da der Parameter Max Informationen aus der Prozedur zurückübermitteln soll, muß er Referenzparameter sein.
Die Prozedur Max2 läßt sich nun wie folgt aufrufen:

```
Max2 (x,y,z)
Max2 (1.5*x,y,z)
Max2 (1.5*x,3,z)
```

Der dritte Parameter von Max2 muß stets eine Variable sein.
Die Prozedur Max2 läßt sich auch als Funktion umschreiben:

```
FUNCTION Max2 (A,B:REAL):REAL;
BEGIN
   IF A>B THEN Max2:= A
                    ELSE Max2:= B
END;
```

Im Funktionskopf werden wieder Name und Parameterliste angegeben. Der Parameter Max fehlt hier. Seine Aufgabe übernimmt der Funktionsname. Die mit dem Doppelpunkt angefügte Typenangabe erklärt ihn zum Datenobjekt vom Typ REAL.

wird fortgesetzt

Hinweis: **In Bild 2.6 des ersten Teils unserer PASCAL-Folge (MP 9/87) sind an Stelle von „T" Apostrophe zu setzen.** ***Red.***

Programmieren in PROLOG

Eine Einführung

Rainer Knauf, Dr. Harald Killenberg
Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik

1. Vorbemerkungen

Die Programmiersprache PROLOG findet seit etwa 1980, als sie in Japan zur Sprache der *Fünften Computer-Generation* erklärt wurde, ein ständig zunehmendes Interesse bei Informatikern und Anwendern. Aufbauend auf der Prädikatenlogik erster Ordnung wurde PROLOG erstmals 1971/1972 von A. Colmerauer in Marseille implementiert.
Im Gegensatz zu den prozeduralen Programmiersprachen (z. B. PASCAL, PL/1, C), in denen beschrieben wird, wie ein Problem zu lösen ist, besteht die Programmierung in der deklarativen Sprache PROLOG in einer logischen Beschreibung eines Sachverhalts (PROLOG = PROgramming in LOGic). Daraus ergibt sich eine prinzipiell andere Herangehensweise beim Entwurf von Programmen, als das bei den prozeduralen Sprachen der Fall ist.
Für eine Reihe von Problemstellungen, insbesondere für komplexe „wissensbasierte" Anwendungen, erweist sich PROLOG nach den Erfahrungen der Autoren als besonders leistungsfähige Programmiersprache /1/.

2. Logische Grundlagen /2/ /3/

Der Grundgedanke der Logik-Programmierung besteht darin, daß das Wissen über einen Gegenstandsbereich als Menge logischer Aussagen formuliert werden kann. Die Grundlage dafür bildet der *Prädikatenkalkül der ersten Stufe.* Diese Aussagen sollen wahre Sachverhalte (Tatsachen) repräsentieren. Sie werden in Form sogenannter *Hornklauseln,* das sind prädikatenlogische Ausdrücke der Form

$$B \leftarrow \bigwedge_{i \leqq m} A_i$$

notiert. B heißt *Klauselkopf,* die A_i bilden den *Klauselkörper.* B und die A_i sind *Atomformeln,* das heißt elementare (nicht logisch zusammengesetzte) Ausdrücke der Form ⟨Prädikatenname⟩ (⟨Term⟩,⟨Term⟩,...). Die *Terme* ⟨Term⟩ symbolisieren dabei Objekte des Gegenstandsbereiches und ⟨Prädikatenname⟩ ein *Prädikat* (eine Relation, eine Beziehung) über diesen Objekten. Ein n-stelliges Prädikat ist eine Abbildung einer Menge von n-Tupeln von Objekten in die Menge {wahr, falsch}. In den Termen auftretende Variablen gelten bezüglich der ganzen Hornklausel als allqualifiziert.
Ist die Anzahl der Prämissen A_i einer Hornklausel null, das heißt der Klauselkörper leer, so spricht man von Fakten (elementaren Aussagen), anderenfalls von Regeln (logisch zusammengesetzten Aussagen, wenn-dann-Beziehungen). Die als Fakten und Regeln formulierten Aussagen bilden eine *Wissensbasis.*
Anhand dieser Aussagen kann man nun von einer Hypothese (Aussage, von der noch nicht bekannt ist, ob sie wahr oder falsch ist) ermitteln, ob sie Schlußfolgerung der in der Wissensbasis abgelegten Fakten und Regeln ist. Eine Hypothese wird in PROLOG als *Frage* formuliert und setzt sich aus logisch UND-verknüpften Atomformeln zusammen.
Ist beispielsweise $\{K_1,.., K_n\}$ die Menge der in einer Wissensbasis enthaltenen Hornklauseln (kurz: Klauseln) und H eine Hypothese, so wird von PROLOG überprüft, ob

$$\bigwedge_{i \leqq n} K_i \rightarrow H$$

eine allgemeingültige (immer wahre) Aussage ist. Das dazu von PROLOG verwendete Prinzip ist das *Resolutionsprinzip.*

3. Formulierung einer Wissensbasis

Die Formulierung eines Sachverhaltes in PROLOG soll an einem einfachen Beispiel, welches /4/ entnommen wurde, erläutert werden.
Gegeben seien folgende Aussagen:

- Die Kollegen Müller und Meier arbeiten im Raum 1, Knauf im Raum 2, Otto und Schulze im Raum 3.
- Telefone befinden sich in den Räumen 2 und 3.
- Ein Kollege ist anrufbar, wenn er in einem Raum arbeitet, in dem sich ein Telefon befindet.
- Zwei Kollegen können ein Gespräch miteinander führen, wenn sie in demselben Zimmer arbeiten oder beide anrufbar sind.

Es könnten dazu z. B. folgende Prädikate eingeführt werden:

arbeitet_in (K,R):
  Ein Kollege K arbeitet im Raum R.
telefon_in (R):
  Im Raum R befindet sich ein Telefon.
anrufbar (K):
  Ein Kollege K ist anrufbar.
gespräch_möglich(K1,K2):
  Zwei Kollegen K1 und K2 können ein Gespräch miteinander führen.

Mit Hilfe dieser Prädikate läßt sich obiger Sachverhalt in PROLOG etwa so notieren:

```
(1)  arbeitet_in(müller,raum_1).
(2)  arbeitet_in(meier,raum_1).
(3)  arbeitet_in(knauf,raum_2).
(4)  arbeitet_in(otto,raum_3).
(5)  arbeitet_in(schulze,raum_3).
(6)  telefon_in(raum_3).
(7)  Telefon_in(raum_2).
(8)  anrufbar(K):-
          arbeitet_in(K,Raum),
          telefon_in(Raum).
(9)  gespräch_möglich(K1,K2):-
          arbeitet_in(K1,Raum),
          arbeitet_in(K2,Raum).
(10) gespräch_möglich(K1,K2):-
          anrufbar(K1),
          anrufbar(K2).
```

Für die Syntax gelten nach /5/ (die dort beschriebene Syntax hat sich inzwischen zum Quasi-Standard entwickelt) folgende Konventionen (vereinfacht):
Objekte werden u. a. durch *Konstanten,* das heißt *Namen* (müller, raum_2,...) oder *Zahlen* (0,100,−6,...) repräsentiert. Namen sind Zeichenfolgen, beginnend mit einem Kleinbuchstaben, die Buchstaben, Ziffern sowie das Unterstreichungszeichen enthalten können. Weitere mögliche Darstellungen für Objekte sind Listen und strukturierte Terme.
PROLOG-*Variablen* sind Zeichenfolgen, die die gleichen Zeichen wie Namen enthalten, aber mit einem Großbuchstaben oder dem Unterstreichungszeichen beginnen. Innerhalb einer Klausel stehen gleiche Variablen auch für das gleiche Objekt.
Prädikatennamen haben gleiche Syntax wie Namen.
Klauseln werden mit einem Punkt (".") abgeschlossen. Als *Konjunktionssymbol* wird das Komma (",") verwendet. Als *Implikationssymbol* notiert man ":–" bei Regeln. Fakten haben kein Implikationssymbol, der Atomformel folgt sofort der abschließende Punkt.

4. Formulierung und Abarbeitung von Fragen

PROLOG ist nun in der Lage, anhand der Wissensbasis Fragen zu beantworten, zum Beispiel:

Befindet sich im Raum 1 ein Telefon?
?–telefon_in(raum_1).
  nein
Sind in den Räumen 2 und 3 Telefone?
?–telefon_in(raum_2),telefon_in(raum_3).
  ja
Welcher Kollege ist anrufbar?
?–anrufbar(Kollege).
  Kollege = knauf

Wenn die Frage positiv beantwortet werden konnte, wird mit „ja" geantwortet bzw. mit der zuerst gefundenen Variablenbelegung, für die die formulierte Hypothese wahr ist. Anderenfalls wird mit „nein" geantwortet. Eine Frage (ein *Ziel*) beginnt mit „?–". Danach folgen, durch Kommata getrennt, die UND-verknüpften Atomformeln (*Teilziele*). Die Frage wird mit einem Punkt (".") abgeschlossen.
Die Abarbeitung letzterer Frage wird im Bild 1 mit Hilfe eines Suchbaumes veranschaulicht.
Die Teilziele werden in der Reihenfolge abgearbeitet, wie sie in der Frage notiert sind. Bei der Abarbeitung eines Teilziels wird die Wissensbasis von oben nach unten nach einer Klausel durchsucht, deren Kopf mit dem Teilziel *unifizierbar,* das heißt „syntaktisch gleichzumachen" ist. Notwendige Bedingung für den Erfolg der Unifikation ist, daß Prädikatenname und Stelligkeit beider Atomformeln gleich und die einander entsprechenden Terme ihrerseits miteinander unifizierbar sind. Dabei können, vereinfacht betrachtet, folgende Fälle auftreten:

1. Beide Terme sind Konstanten.
Die Unifikation ist erfolgreich, falls die Konstanten identisch sind.
2. Ein Term ist eine Konstante, ein Term ist eine Variable. Die Unifikation ist erfolgreich, die Variable wird durch die Konstante ersetzt.
3. Beide Terme sind Variablen.
Die Unifikation ist erfolgreich. Die Variablen stehen nun für das gleiche Objekt.

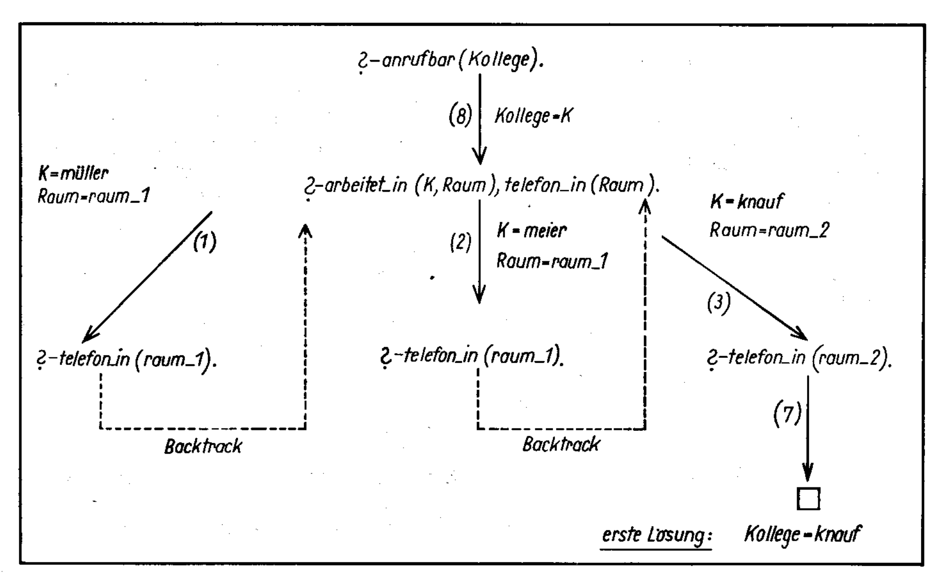


Bild 1 Suchbaum bei der Abarbeitung von „?-anrufbar(Kollege)".

Dipl.-Ing. Rainer Knauf (24) studierte von 1982 bis 1987 an der Technischen Hochschule Ilmenau in der Fachrichtung Technische Kybernetik und Automatisierungstechnik mit Spezialisierung Technische Informatik. Er diplomierte im Februar 1987 mit einem Thema über Expertensysteme und deren Implementierung in PROLOG. Seit März dieses Jahres ist er wissenschaftlicher Assistent des Bereiches Technische Informatik.

Doz. Dr. sc. techn. Harald Killenberg erwarb 1969 das Diplom an der Technischen Hochschule Ilmenau, Fachrichtung Technische Kybernetik und Automatisierungstechnik; 1974 und 1983 Promotionen zum Dr.-Ing. und Dr. sc. techn. mit Arbeiten zur Beschreibung und einem Entwurf industrieller Steuerungssysteme; 1977–1978 Zusatzstudium am Leningrader Institut für Feinmechanik und Optik; 1984 Berufung zum Hochschuldozenten für das Gebiet Nichtnumerische Informationsverarbeitung/Künstliche Intelligenz; tätig als Projektleiter im VEB Robotron-Elektronik Zella-Mehlis (1984–1986) und als Hochschullehrer im Wissenschaftsbereich Technische Informatik der Sektion Technische und Biomedizinische Kybernetik der Technischen Hochschule Ilmenau.

War die Unifikation eines Teilziels mit einem Klauselkopf erfolgreich, so wird im Ziel dieses Teilziel durch den zugehörigen Klauselkörper ersetzt und das so entstandene aktuelle Ziel auf gleiche Weise weiterbearbeitet. Ein solches Verfahren heißt *Tiefensuche*. Bei der Unifikation eines Teilziels mit einem Fakt wird dieses, da der Klauselkörper leer ist, aus dem aktuellen Ziel entfernt. Die sich bei der Unifikation ergebenden Variablenbelegungen (der Unifikator) werden in das entstandene Ziel übernommen.
Bei dem im Bild dargestellten Beispiel bilden die Knoten des Suchbaumes das jeweils aktuelle Ziel. Die Kanten sind mit der Nummer der jeweils zur Unifikation angewandten Klausel sowie dem Unifikator markiert. Stößt das System auf alternativ anwendbare Klauseln (d. h. auf eine „Weggabelung" im Suchbaum), so richtet es sich einen Entscheidungspunkt (place marker) ein und merkt sich den momentanen Abarbeitungszustand (die Variablenbelegungen). Es wird zunächst die erste der alternativen Klauseln in der Wissensbasis angewandt.
Bleibt eine Unifikation erfolglos, so wird ein *Backtrack* ausgelöst, das heißt, der Algorithmus setzt zum letzten vorausgegangenen Entscheidungspunkt zurück und führt, ausgehend vom dort gekellerten Zustand, die Bearbeitung mit der nächsten alternativen Klausel fort. Die Abarbeitung wird
– erfolgreich beendet (Antwort „ja" bzw. eine Variablenbelegung), wenn das aktuelle Ziel leer ist (Symbol "□" im Bild)
– erfolglos beendet (Antwort „nein"), wenn für ein Teilziel keine Klausel mehr zur Unifikation angewandt werden kann. Nach einer erfolgreichen Beendigung der Abarbeitung kann auf Wunsch ein Backtrack erzwungen werden, so daß nach weiteren Lösungen gesucht wird.
Aus dem von PROLOG verwendeten Suchverfahren (Tiefensuche mit Backtrack) ergeben sich folgende Konsequenzen:
1. Die Reihenfolge der Klauseln in der Wissensbasis ist für die Lösung von Bedeutung. Wären z. B. die Klauseln (3) und (5) vertauscht, ergäbe sich im Beispiel als erste Lösung „Kollege = schulze".
2. Die Reihenfolge der Teilziele in Klauselkörpern ist für die Lösung von Bedeutung. Wäre z. B. diese Reihenfolge in (8) umgekehrt, ergäbe sich im Beispiel als erste Lösung „Kollege = otto".
3. Das Verfahren terminiert nicht zwangsläufig.

5. Listen und Rekursionen

Eine in PROLOG oft verwendete Datenstruktur ist die *Liste*.
Definition:

1. [ ] ist eine Liste (die *leere Liste*).
2. Wenn T ein Term und L eine Liste oder eine Variable ist, dann ist [T|L] eine Liste.

"|", der Listenabtrennoperator, teilt die Liste in *Listenkopf* (links von "|") und *Listenkörper* bzw. *Restliste* (rechts von "|").
Alternative Schreibweisen ein und derselben Liste sind

$[T_1 \mid [T_2 \mid [T_3 \mid []]]]$
$[T_1 \mid [T_2 \mid [T_3]]]$
$[T_1 \mid [T_2, T_3]]$
$[T_1, T_2, T_3]$

Die Terme T_i heißen *Listenelemente.* Mit Hilfe von Listen könnte man z. B. die Unterstellungsverhältnisse in einem Betrieb etwa so darstellen:

```
chef_von([müller,knauf,schulze],meier).
chef_von([lehmann,walter],otto).
chef_von([meier,otto],albrecht).
```

Ein in PROLOG vorrangiger Programmierstil ist die *Rekursion.* Dabei handelt es sich um den Selbstaufruf des Kopfprädikates in Regelkörpern. An einem einfachen Beispiel aus der Listenverarbeitung, der Mitgliedschaft eines Elementes in einer Liste, soll die Rekursion erläutert werden.

```
mitglied(E,[E|L]).
mitglied(E1,[E2|L]):–
mitglied(E1,L).
```

„mitglied" ist ein 2stelliges Prädikat und soll dann „wahr" sein, wenn das erste Argument ein Element der als zweites Argument stehenden Liste ist. Zunächst wird der einfachste Fall einer solchen Mitgliedschaft als Fakt notiert: Ein Element E ist dann Mitglied in einer Liste, wenn es im Listenkopf steht, d. h. die Liste die Struktur [E|L] hat. L ist dabei eine beliebige Restliste.
Ist dieser Fall für ein konkretes Ziel nicht zutreffend, so wird, da ein Backtrack ausgelöst wird, die zweite Klausel angewandt. Hier wird die Mitgliedschaft eines Elements in einer Liste auf die Mitgliedschaft in der Restliste zurückgeführt. Der Selbstaufruf von „mitglied" wird so lange fortgeführt, bis durch das wiederholte Abspalten des ersten Elements
– die erste Klausel zutreffend ist (Ziel erfolgreich bearbeitet) oder
– die leere Liste entstanden ist, so daß keine der beiden Klauseln mehr anwendbar ist (Ziel erfolglos bearbeitet).

6. Eingebaute Prädikate

Es gibt einige Prädikate, die häufig gebraucht werden und zum Teil mit Hilfe von Fakten und Regeln nur umständlich oder gar nicht definiert werden können. Derartige Prädikate sind oft bereits vordefiniert und werden als *eingebaute Prädikate* bezeichnet. Anzahl und Sortiment eingebauter Prädikate ist bei verschiedenen PROLOG-Übersetzern zum Teil recht unterschiedlich. Es handelt sich i. allg. um Prädikate

– zum Test von Termen (ob Variable, Konstante, Liste, ...)
– zum Vergleich von Termen
– zur Arithmetik
– zur Ein- und Ausgabe
– zur Steuerung der Abarbeitung (Manipulation des Abarbeitungsmechanismus) und
– zur Veränderung der Wissensbasis während der Programmabarbeitung.

Eine Aufstellung dieser Prädikate ist dem Beschreibungsmaterial des jeweils verwendeten Interpreters bzw. Compilers zu entnehmen.

Literatur und Kontaktadresse auf Seite 341

Literatur

/1/ Knauf, R.: Wissensverarbeitung mit der Sprache PROLOG am Beispiel eines Diagnosesystems für technische Geräte. Mikroprocessory – dydaktika i zastosowania, VII. und VIII. wissenschaftliches Seminar, Zielona Gora, 1987, S. 114–120

/2/ Asser, G.: Einführung in die mathematische Logik. Teil 1: Aussagenkalkül, 1982, Teil 2: Prädikatenkalkül der ersten Stufe, 1975; BSB B.G. Teubner Verlagsgesellschaft

/3/ Gürtzig, K.: Untersuchungen zur Anwendung der Logik-Programmierung für Probleme der Steuerungstechnik. Dissertation A, TH Ilmenau, 1987

/4/ Gürztig, K.; Schumann, H.-D.: E-PROLOG – ein einfacher Experimental-Prolog-Interpreter für die Ausbildung. Wissenschaftliche Zeitschrift der TH Ilmenau 32 (1986) 3, S. 71 ff

/5/ Clocksin, W. F.; Mellish, C. S.: Programming in Prolog. Springer-Verlag, Berlin–Heidelberg–New York, 1981

/6/ Kleine Büning, H.; Schmitgen, S.: Prolog. B.G. Teubner, Stuttgart, 1986

/7/ Schnupp, P.: PROLOG – Einführung in die Programmierpraxis. Hanser-Verlag, München Wien, 1986

✉ KONTAKT ☏

Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik, Wissenschaftsbereich Technische Informatik, PSF 327, Ilmenau, 6300; Tel. 74561, R. Knauf

Disassembler für den KC85/2 (/3)

Lutz Molgedey, Berlin

Will man ein bereits in Maschinensprache geschriebenes Programm analysieren, so stehen dafür nicht immer umfangreiche Programme zur Verfügung, oder diese sind einfach zu lang bzw. stören sich mit dem zu analysierenden Programm. Oft werden auch vom Hersteller nicht definierte Befehle /2, 3/ verwendet. Diese können die meisten herkömmlichen Disassembler nicht übersetzen. Deshalb wurde ein im folgenden beschriebener Disassembler für den KC 85/2 (/3) entwikkelt, der jeder Bytefolge eine Mnemonik zuordnet, ohne das Maschinenprogramm weiter zu analysieren. Es wird also kein Quelltext und keine Symboltabelle erzeugt, sondern der Anwender muß selber entscheiden, ob bestimmte Byte als Befehle oder Daten zu interpretieren sind.

Der Disassembler beginnt bei BA00 und ist etwas länger als 0.75 KByte. Arbeitsplatz im RAM wird dabei nicht benötigt. Er ist ohne größere Probleme auf jede integrale 100-Grenze verschiebbar; dazu müssen die im Programm (Bild 1) hervorgehobenen Bytes geändert werden. Aus dem Menü erfolgt der Aufruf durch **DIS adr. oder DIS adr. ENTER.** Mit der Taste **Break** wird das Programm verlassen, jede andere Taste bewirkt das Auslisten der nächsten Zeile. Der Disassembler führt keine Modulschaltungen durch, der IRM ist also immer eingeschaltet. Damit können in dieser Version Programme, die im Schatten des Bildschirmwiederholspeichers liegen, nicht übersetzt werden.

Nach dem Aufruf werden als erstes die Vorbytes DD bzw. FD abgesucht, dabei erhält nur das letzte Bedeutung. Danach wird nach den Vorbytes CB und ED gefragt, das Auftreten von ED macht die Vorbytes DD bzw. FD wirkungslos. Nun wird das nächste Code-Byte mit Masken belegt (AND-Befehle) und dadurch der Befehl festgestellt. Danach wird noch lediglich die entsprechende Mnemonik aus der Tabelle herausgesucht und ausgegeben.

Die Befehlsdarstellung weicht in einigen Fällen von der üblichen U880-Mnemonik /1/ ab, z. B. CP statt CMP und JP statt JMP. Die Darstellung der zusätzlichen Befehle entspricht der in /2/ angegebenen. Für den Befehl ED71 wurde die Mnemonik OTCLR verwendet. Bei Relativsprüngen wird die absolute Zieladresse angegeben.

Der Disassembler besteht aus zwei wesentlichen Teilen, als erstes aus dem Tabellenteil und dann als zweites aus dem eigentlichen Programm. Innerhalb des Tabellenteils kommt als erstes die Maskentabelle und danach die Mnemoniktabelle. Im Programm selbst werden die Befehle auf die Grundbefehle zurückgeführt. Das betrifft vor allem die Arbeit mit den Indexregistern. Dazu werden im Register C (Tafel 2) Flags gesetzt, und mit ihnen wird dann die weitere Übersetzung gesteuert. Um Speicherplatz zu sparen, wird im Programm ein eigener Zeichensatz (Tafel 1) verwendet, dieser wird nur bei der Ausgabe in den ASCII-Satz umgerechnet. Will man eine andere Mnemonik verwenden, z. B. die Z80-Mnemonik, so muß dementsprechend die Mnemoniktabelle geändert werden. Soll beispielsweise JP cc,nn statt JPcc nn verwendet werden, so muß in der Tabelle D575223726 statt D55B2226 stehen. Dazu muß dann jedoch bei den Bedingungszeichen (NZ...M) das SPACE weggenommen werden (in der Tabelle ab BAC3). Eventuell kann man dann noch die Ausgabe von SPACE (bei BC3E) durch einen Tabulator ersetzen.

Tafel 1 Interner Zeichencode

	Ø/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
Ø	LD␣	SBC␣	rot	%	5	E	U	K␣
1	RET	AND␣	r	&	6	F	V	L␣
2	NOP	XOR␣	cc␣	'	7	G	W	M␣
3	OUT␣	OR␣	cc'␣	(	8	H	X	N␣
4	INC␣	CP␣	blo	)	9	I	Y	O␣
5	DEC␣	BC	n	*	:	J	Z	P␣
6	(nn)	BC	nn	+	;	K	A␣	Q␣
7	(dd)	DE	e	,	<	L	B␣	R␣
8	dd,	DE	b	-	=	M	C␣	S␣
9	,dd	HL	r'	.	>	N	D␣	T␣
A	A,	HL	r'_{CB}	/	?	O	E␣	U␣
B	,A	AF	␣	Ø	a	P	F␣	V␣
C	b,r'	SP	!	1	A	Q	G␣	W␣
D	ADD␣	ari␣	"	2	B	R	H␣	X␣
E	ADC␣	qq	#	3	C	S	I␣	Y␣
F	SUB␣	dd	$	4	D	T	J␣	Z␣

␣ =SPACE
r =8-Bit-Register (Bit 345)
r' =8-Bit-Register (Bit Ø12)
dd =16-Bit-Register (BC DE HL SP)
qq =16-Bit-Register (BC DE HL AF)
n =8-Bit-Zahl
nn =16-Bit-Zahl
e =8-Bit-Distanzadresse
b =Bitposition (Ø...7)
cc =Bedingung (NZ...M)
cc'=Bedingung (NZ...C)
ari=Arithmetik (ADD...CP)
rot=Rotation (RLC...SRL)
blo=Block (I D IR DR)

Tafel 2 Das Register C wird als Flag-Register benutzt

```
Register C
      Bit 7 - 1=CB-Befehl + Vorbyte DD/FD
      Bit 6 - 1=(IX+d)/(IY+d) bereits aufgetreten
      Bit 5 -
      Bit 4 -
      Bit 3 -
      Bit 2 -
      Bit 1 - 1=DD/FD-Befehl
      Bit Ø - Ø=Vorbyte DD
              1=Vorbyte FD
```

```
                    ;Disassembler fuer den KC85/2 (/3)
                    ;
          OS:       EQU    ØFØØ3H   ;Betriebssystem PVI
          CURS:     EQU    ØB7AØH
          BEG:      EQU    Ø5DØ1H   ;Anfangsadresse/2 + 1
                    ORG    ØBAØØH
BAØØ                ;Adressen fuer Masken
BAØØ 6A             DB     Ø6AH     ;Mnemonik-Nr. ED-Vorbyte -1
BAØ1 3A             DB     Ø3AH     ;Maskenadresse ED-Vorbyte
BAØ2 39             DB     Ø39H     ;Mnemonik-Nr. o. Vorbyte -1
BAØ3 Ø6             DB     ØØ6H     ;Maskenadresse o. Vorbyte
BAØ4 66             DB     Ø66H     ;Mnemonik-Nr. CB-Vorbyte -1
BAØ5 35             DB     Ø35H     ;Maskenadresse CB-Vorbyte
BAØ6                ;Maskentabelle ohne Vorbyte
BAØ6 FF             DB     ØFFH     ;Maske
BAØ7 ØØ             DB     ØØØH     ;NOP
BAØ8 1Ø             DB     Ø1ØH     ;DJNZ e
BAØ9 22             DB     Ø22H     ;LD (nn),HL
BAØA 32             DB     Ø32H     ;LD (nn),A
BAØB 27             DB     Ø27H     ;DAA
BAØC 37             DB     Ø37H     ;SCF
BAØD Ø8             DB     ØØ8H     ;EXAF
BAØE 18             DB     Ø18H     ;JR e
BAØF 2A             DB     Ø2AH     ;LD HL,(nn)
BA1Ø 3A             DB     Ø3AH     ;LD A,(nn)
BA11 2F             DB     Ø2FH     ;CPL
BA12 3F             DB     Ø3FH     ;CCF
BA13 76             DB     Ø76H     ;HALT
BA14 C3             DB     ØC3H     ;JP nn
BA15 D3             DB     ØD3H     ;OUT n
BA16 E3             DB     ØE3H     ;EX (SP),HL
BA17 F3             DB     ØF3H     ;DI
BA18 C9             DB     ØC9H     ;RET
BA19 D9             DB     ØD9H     ;EXX
BA1A E9             DB     ØE9H     ;JP (HL)
BA1B F9             DB     ØF9H     ;LD SP,HL
BA1C DB             DB     ØDBH     ;IN n
BA1D EB             DB     ØEBH     ;EX DE,HL
BA1E FB             DB     ØFBH     ;EI
BA1F CD             DB     ØCDH     ;CALL nn
BA2Ø FF             DB     ØFFH     ;Ende-Zeichen
BA21 CF             DB     ØCFH     ;Maske
BA22 Ø1             DB     ØØ1H     ;LD dd,nn
BA23 Ø2             DB     ØØ2H     ;LD (dd),A
BA24 Ø3             DB     ØØ3H     ;INC dd
BA25 Ø9             DB     ØØ9H     ;ADD HL,dd
BA26 ØA             DB     ØØAH     ;LD A,(dd)
BA27 ØB             DB     ØØBH     ;DEC dd
BA28 C1             DB     ØC1H     ;POP qq
BA29 C5             DB     ØC5H     ;PUSH qq
BA2A FF             DB     ØFFH     ;Ende-Zeichen
BA2B C7             DB     ØC7H     ;Maske
BA2C ØØ             DB     ØØØH     ;JRcc' e
BA2D Ø4             DB     ØØ4H     ;INC r
BA2E Ø5             DB     ØØ5H     ;DEC r
BA2F Ø6             DB     ØØ6H     ;LD r,n
BA3Ø CØ             DB     ØCØH     ;Rcc
BA31 C2             DB     ØC2H     ;JPcc nn
BA32 C4             DB     ØC4H     ;CAcc nn
BA33 C6             DB     ØC6H     ;ADD n ... CP n
```

```
BA34 FF             DB     ØFFH     ;Ende-Zeichen
BA35                ;Maskentabelle ohne Vorbyte / CB-Vorbyte
BA35 CØ             DB     ØCØH     ;Maske
BA36 ØØ             DB     ØØØH     ;RLCA...RRA /RLC r'...SRL r'
BA37 4Ø             DB     Ø4ØH     ;LD r,r'    /BIT b,r'
BA38 8Ø             DB     Ø8ØH     ;ADD r'..CP r'/RES b,r'
BA39 CØ             DB     ØCØH     ;RSTØ...RST7/SET b,r'
BA3A                ;Maskentabelle ED-Vorbyte
BA3A FF             DB     ØFFH     ;Maske
BA3B 7Ø             DB     Ø7ØH     ;INF
BA3C 71             DB     Ø71H     ;OTCLR
BA3D 47             DB     Ø47H     ;LD I,A
BA3E 57             DB     Ø57H     ;LD A,I
BA3F 67             DB     Ø67H     ;RRD
BA4Ø 4D             DB     Ø4DH     ;RETI
BA41 4F             DB     Ø4FH     ;LD R,A
BA42 5F             DB     Ø5FH     ;LD A,R
BA43 6F             DB     Ø6FH     ;RLD
BA44 FF             DB     ØFFH     ;Ende-Zeichen
BA45 CF             DB     ØCFH     ;Maske
BA46 42             DB     Ø42H     ;SBC HL,dd
BA47 43             DB     Ø43H     ;LD (nn),dd
BA48 4A             DB     Ø4AH     ;ADC HL,dd
BA49 4B             DB     Ø4BH     ;LD dd,(nn)
BA4A FF             DB     ØFFH     ;Ende-Zeichen
BA4B C7             DB     ØC7H     ;Maske
BA4C 4Ø             DB     Ø4ØH     ;IN r
BA4D 41             DB     Ø41H     ;OUT r
BA4E 44             DB     Ø44H     ;NEG
BA4F 45             DB     Ø45H     ;RETN
BA5Ø FF             DB     ØFFH     ;Ende-Zeichen
BA51 E7             DB     ØE7H     ;Maske
BA52 AØ             DB     ØAØH     ;LDI/LDIR/LDD/LDDR
BA53 A1             DB     ØA1H     ;CPI/CPIR/CPD/CPDR
BA54 A2             DB     ØA2H     ;INI/INIR/IND/INDR
BA55 A3             DB     ØA3H     ;OTI/OTIR/OTD/OTDR
BA56 FF             DB     ØFFH     ;Ende-Zeichen
BA57 D7             DB     ØD7H     ;Maske
BA58 46             DB     Ø46H     ;IMØ
BA59 FF             DB     ØFFH     ;Ende-Zeichen
BA5A DF             DB     ØDFH     ;Maske
BA5B 56             DB     Ø56H     ;IM1
BA5C 5E             DB     Ø5EH     ;IM2
BA5D FF             DB     ØFFH     ;Ende-Zeichen
BA5E ØØ             DB     ØØØH     ;Maske
BA5F ØØ   BMNE:     DB     ØØØH     ;NOP
BA6Ø                ;Macro-Tabelle
BA6Ø D769           DB     ØD7H,69H          ;LD␣      (ØØ)
BA62 DD5Ø5F         DB     ØDDH,5ØH,5FH      ;RET      (Ø1)
BA65 D95A5B         DB     ØD9H,5AH,5BH      ;NOP      (Ø2)
BA68 DA6Ø79         DB     ØDAH,6ØH,79H      ;OUT␣     (Ø3)
BA6B D45968         DB     ØD4H,59H,68H      ;INC␣     (Ø4)
BA6E CF5Ø68         DB     ØCFH,5ØH,68H      ;DEC␣     (Ø5)
BA71 B32634         DB     ØB3H,26H,34H      ;(nn)     (Ø6)
BA74 B31F34         DB     ØB3H,1FH,34H      ;(dd)     (Ø7)
BA77 9F37           DB     Ø9FH,37H          ;dd,      (Ø8)
BA79 B71F           DB     ØB7H,1FH          ;,dd      (Ø9)
BA7B CC37           DB     ØCCH,37H          ;A,       (ØA)
BA7D B74C           DB     ØB7H,4CH          ;,A       (ØB)
BA7F A83729         DB     ØA8H,37H,29H      ;b,r'     (ØC)
```

```
BA82 CC4F69         DB     ØCCH,4FH,69H      ;ADD␣     (ØD)
BA85 CC4F68         DB     ØCCH,4FH,68H      ;ADC␣     (ØE)
BA88 DE6Ø67         DB     ØDEH,6ØH,67H      ;SUB␣     (ØF)
BA8B DE4D68         DB     ØDEH,4DH,68H      ;SBC␣     (1Ø)
BA8E CC5969         DB     ØCCH,59H,69H      ;AND␣     (11)
BA91 E313           DB     ØE3H,13H          ;XOR␣     (12)
BA93 DA77           DB     ØDAH,77H          ;OR␣      (13)
BA95 CE75           DB     ØCEH,75H          ;CP␣      (14)
BA97 96             DB     Ø96H              ;BC       (15)
BA98 CD4E           DB     ØCDH,4EH          ;BC       (16)
BA9A 98             DB     Ø98H              ;DE       (17)
BA9B CF5Ø           DB     ØCFH,5ØH          ;DE       (18)
BA9D 9A             DB     Ø9AH              ;HL       (19)
BA9E D357           DB     ØD3H,57H          ;HL       (1A)
BAAØ CC51           DB     ØCCH,51H          ;AF       (1B)
BAA2 DE5B           DB     ØDEH,5BH          ;SP       (1C)
BAA4 DD574E         DB     ØDDH,57H,4EH      ;RLC      (1D)
BAA7 DD5D4E         DB     ØDDH,5DH,4EH      ;RRC      (1E)
BAAA DD57           DB     ØDDH,57H          ;RL       (1F)
BAAC DD5D           DB     ØDDH,5DH          ;RR       (2Ø)
BAAE DE574C         DB     ØDEH,57H,4CH      ;SLA      (21)
BAB1 DE5D4C         DB     ØDEH,5DH,4CH      ;SRA      (22)
BAB4 DE574C54       DB     ØDEH,57H,4CH,54H  ;SLAI     (23)
BAB8 DE5D57         DB     ØDEH,5DH,57H      ;SRL      (24)
BABB CD             DB     ØCDH              ;B        (25)
BABC CE             DB     ØCEH              ;C        (26)
BABD CF             DB     ØCFH              ;D        (27)
BABE DØ             DB     ØDØH              ;E        (28)
BABF D3             DB     ØD3H              ;H        (29)
BACØ D7             DB     ØD7H              ;L        (2A)
BAC1 D8             DB     ØD8H              ;M        (2B)
BAC2 CC             DB     ØCCH              ;A        (2C)
BAC3 D97F           DB     ØD9H,7FH          ;NZ␣      (2D)
BAC5 FF             DB     ØFFH              ;Z␣       (2E)
BAC6 D968           DB     ØD9H,68H          ;NC␣      (2F)
BAC8 E8             DB     ØE8H              ;C␣       (3Ø)
BAC9 DB74           DB     ØDBH,74H          ;PO␣      (31)
BACB DB6A           DB     ØDBH,6AH          ;PE␣      (32)
BACD F5             DB     ØF5H              ;P␣       (33)
BACE F2             DB     ØF2H              ;M␣       (34)
BACF D4             DB     ØD4H              ;I        (35)
BADØ CF             DB     ØCFH              ;D        (36)
BAD1 D45D           DB     ØD4H,5DH          ;IR       (37)
BAD3 CF5D           DB     ØCFH,5DH          ;DR       (38)
BAD5 B31A362534     DB     ØB3H,1AH,36H,25H,34H ;(IX+d) bzw.
BADA                                     ;(IY+d)        (39)
BADA                ;Mnemonik-Tabelle ohne Vorbyte
BADA 82             DB     Ø82H              ;NOP           (3A)
BADB CF55597F27     DB     ØCFH,55H,59H,7FH,27H ;DJNZ e     (3B)
BAEØ 8ØØ6Ø9         DB     Ø8ØH,Ø6H,Ø9H      ;LD (nn),HL    (3C)
BAE3 8ØØ6ØB         DB     Ø8ØH,Ø6H,ØBH      ;LD (nn),A     (3D)
BAE6 CF4C4C         DB     ØCFH,4CH,4CH      ;DAA           (3E)
BAE9 DE4E51         DB     ØDEH,4EH,51H      ;SCF           (3F)
BAEC DØ631B         DB     ØDØH,63H,1BH      ;EXAF          (4Ø)
BAEF D57727         DB     ØD5H,77H,27H      ;JR e          (41)
BAF2 8ØØ8Ø6         DB     Ø8ØH,Ø8H,Ø6H      ;LD HL,(nn)    (42)
BAF5 8ØØAØ6         DB     Ø8ØH,ØAH,Ø6H      ;LD A,(nn)     (43)
BAF8 CE5B57         DB     ØCEH,5BH,57H      ;CPL           (44)
BAFB CE4E51         DB     ØCEH,4EH,51H      ;CCF           (45)
BAFE D34C575F       DB     ØD3H,4CH,57H,5FH  ;HALT          (46)
```

```
BBØ2 D57526         DB     ØD5H,75H,26H      ;JP nn         (47)
BBØ5 8325           DB     Ø83H,25H          ;OUT n         (48)
BBØ7 DØ7D331C34Ø9   DB     ØDØH,7DH,33H,1CH,34H,Ø9H
BBØD                                         ;EX (SP),HL    (49)
BBØD CF54           DB     ØCFH,54H          ;DI            (4A)
BBØF 81             DB     Ø81H              ;RET           (4B)
BB1Ø DØ6363         DB     ØDØH,63H,63H      ;EXX           (4C)
BB13 D575Ø7         DB     ØD5H,75H,Ø7H      ;JP (HL)       (4D)
BB16 8ØØ81A         DB     Ø8ØH,Ø8H,1AH      ;LD SP,HL      (4E)
BB19 D47325         DB     ØD4H,73H,25H      ;IN n          (4F)
BB1C DØ7D18375357   DB     ØDØH,7DH,18H,37H,53H,57H
BB22                                         ;EX DE,HL      (5Ø)
BB22 DØ54           DB     ØDØH,54H          ;EI            (51)
BB24 CE4C577126     DB     ØCEH,4CH,57H,71H,26H ;CALL nn    (52)
BB29 8ØØ826         DB     Ø8ØH,Ø8H,26H      ;LD dd,nn      (53)
BB2C 8ØØ7ØB         DB     Ø8ØH,Ø7H,ØBH      ;LD (dd),A     (54)
BB2F 841F           DB     Ø84H,1FH          ;INC dd        (55)
BB31 8D1AØ9         DB     Ø8DH,1AH,Ø9H      ;ADD HL,dd     (56)
BB34 8ØØAØ7         DB     Ø8ØH,ØAH,Ø7H      ;LD A,(dd)     (57)
BB37 851F           DB     Ø85H,1FH          ;DEC dd        (58)
BB39 DB5A751E       DB     ØDBH,5AH,75H,1EH  ;POP qq        (59)
BB3D DB6Ø5E6D1E     DB     ØDBH,6ØH,5EH,6DH,1EH ;PUSH qq    (5A)
BB42 D55D2327       DB     ØD5H,5DH,23H,27H  ;JRcc' e       (5B)
BB46 8421           DB     Ø84H,21H          ;INC r         (5C)
BB48 8521           DB     Ø85H,21H          ;DEC r         (5D)
BB4A 8Ø213725       DB     Ø8ØH,21H,37H,25H  ;LD r,n        (5E)
BB4E DD22           DB     ØDDH,22H          ;Rcc           (5F)
BB5Ø D55B2226       DB     ØD5H,5BH,22H,26H  ;JPcc nn       (6Ø)
BB54 CE4C2226       DB     ØCEH,4CH,22H,26H  ;CAcc nn       (61)
BB58 9D25           DB     Ø9DH,25H          ;ADD n...CP n  (62)
BB5A AØ4C           DB     ØAØH,4CH          ;RLCA...RRA    (63)
BB5C 8Ø213729       DB     Ø8ØH,21H,37H,29H  ;LD r,r'       (64)
BB6Ø 9D29           DB     Ø9DH,29H          ;ADD r'.CP r'  (65)
BB62 DD5E5F28       DB     ØDDH,5EH,5FH,28H  ;RSTØ...RST7   (66)
BB66                ;Mnemonik-Tabelle CB-Vorbyte
BB66 AØ2B292A       DB     ØAØH,2BH,29H,2AH  ;RLC r'.SRL r' (67)
BB6A CD5479ØC       DB     ØCDH,54H,79H,ØCH  ;BIT b,r'      (68)
BB6E DD5Ø78ØC2A     DB     ØDDH,5ØH,78H,ØCH,2AH ;RES b,r'   (69)
BB73 DE5Ø79ØC2A     DB     ØDEH,5ØH,79H,ØCH,2AH ;SET b,r'   (6A)
BB78                ;Mnemonik-Tabelle ED-Vorbyte
BB78 D45951         DB     ØD4H,59H,51H      ;INF           (6B)
BB7B DA5F4E575D     DB     ØDAH,5FH,4EH,57H,5DH ;OTCLR      (6C)
BB8Ø 8Ø54ØB         DB     Ø8ØH,54H,ØBH      ;LD I,A        (6D)
BB83 8ØØA54         DB     Ø8ØH,ØAH,54H      ;LD A,I        (6E)
BB86 DD5D4F         DB     ØDDH,5DH,4FH      ;RRD           (6F)
BB89 8154           DB     Ø81H,54H          ;RETI          (7Ø)
BB8B 8Ø5DØB         DB     Ø8ØH,5DH,ØBH      ;LD R,A        (71)
BB8E 8ØØA54         DB     Ø8ØH,ØAH,54H      ;LD A,R        (72)
BB91 DDØØ           DB     ØDDH,ØØH          ;RLD           (73)
BB93 9Ø1AØ9         DB     Ø9ØH,1AH,Ø9H      ;SBC HL,dd     (74)
BB96 8ØØ6Ø9         DB     Ø8ØH,Ø6H,Ø9H      ;LD (nn),dd    (75)
BB99 8E1AØ9         DB     Ø8EH,1AH,Ø9H      ;ADC HL,dd     (76)
BB9C 8ØØ8Ø6         DB     Ø8ØH,Ø8H,Ø6H      ;LD dd,(nn)    (77)
BB9F D47321         DB     ØD4H,73H,21H      ;IN r          (78)
BBA2 8321           DB     Ø83H,21H          ;OUT r         (79)
BBA4 D95Ø52         DB     ØD9H,5ØH,52H      ;NEG           (7A)
BBA7 8159           DB     Ø81H,59H          ;RETN          (7B)
BBA9 D74F24         DB     ØD7H,4FH,24H      ;LDI/LDIR
BBAC                                         ;LDD/LDDR      (7C)
BBAC CE5B24         DB     ØCEH,5BH,24H      ;CPI/CPIR
```

```
BBAF                                         ;CPD/CPDR   (7D)
BBAF D45924          DB      ØD4H,59H,24H    ;INI/INIR
BBB2                                         ;IND/INDR   (7E)
BBB2 DA5F24          DB      ØDAH,5FH,24H    ;OTI/OTIR
BBB5                                         ;OTD/OTDR   (7F)
BBB5 D4583B          DB      ØD4H,58H,3BH    ;IMØ        (8Ø)
BBB8 D4583C          DB      ØD4H,58H,3CH    ;IM1        (81)
BBBB D4583D          DB      ØD4H,58H,3DH    ;IM2        (82)
BBBE 82              DB      Ø82H            ;NOP        (83)
BBBF                 ;Nukleus
BBBF FECB    NIND:   CP      ØCBH      ;Vorbyte CB ?
BBC1 21Ø15D          LD      HL,BEG    ;Anfangsadresse/2 + 1
BBC4 2Ø1E            JRNZ    NCB
BBC6 2C              INC     L         ;CB-Maskentabelle
BBC7 CB49            BIT     1,C       ;DD/FD ?
BBC9 282Ø            JRZ     ZBYTE
BBCB CBF9            SET     7,C       ;CB + DD/FD
BBCD 13              INC     DE        ;Ueberspringe Ablage
BBCE 181B            JR      ZBYTE
BBDØ ØEØØ    DIS:    LD      C,ØØH     ;Start Nukleus
BBD2 1A      BLE:    LD      A,(DE)    ;hole Byte
BBD3 FEDD            CP      ØDDH      ;Vorbyte DD ?
BBD5 2ØØ4            JRNZ    NDD
BBD7 ØEØ2            LD      C,Ø2H     ;Flag-Reg. auf DD-Vorbyte
BBD9 18Ø6            JR      WE
BBDB FEFD    NDD:    CP      ØFDH      ;Vorbyte FD ?
BBDD 2ØBØ            JRNZ    NIND
BBDF ØEØ3            LD      C,Ø3H     ;Flag-Reg. auf FD-Vorbyte
BBE1 13      WE:     INC     DE        ;Zeiger erhoehen
BBE2 18EE            JR      BLE
BBE4 FEED    NCB     CP      ØEDH      ;Vorbyte ED ?
BBE6 2ØØ4            JRNZ    NED
BBE8 2D              DEC     L         ;ED-Maskentabelle
BBE9 ØEØØ            LD      C,ØØH     ;DD/FD rueckgaengig
BBEB 13      ZBYTE:  INC     DE        ;Zeiger auf Code-Byte
BBEC 29      NED:    ADD     HL,HL
BBED 46              LD      B,M       ;Mnemonik-Nr. - 1
BBEE 23              INC     HL
BBEF 6E              LD      L,M       ;Maskenadresse
BBFØ 1A      SUCH:   LD      A,(DE)    ;hole Code-Byte
BBF1 A6              AND     M         ;Maske
BBF2 23              INC     HL
BBF3 FD6F            LD      LY,A
BBF5 7E      CMP:    LD      A,M       ;Befehl in Teiltabelle
BBF6 23              INC     HL        ;enthalten
BBF7 FEFF            CP      ØFFH      ;Ende-Zeichen Teiltabelle ?
BBF9 28F5            JRZ     SUCH
BBFB Ø4              INC     B         ;erhoehe Mnemonik-Nr.
BBFC FDBD            CP      LY        ;Befehl gefunden ?
BBFE 2ØF5            JRNZ    CMP       ;sonst weitersuchen
BCØØ C5              PUSH    BC
BCØ1 1A              LD      A,(DE)    ;hole Code-Byte
BCØ2 13              INC     DE        ;Zeiger auf naechstes Byte
BCØ3 FD6F            LD      LY,A      ;Code-Byte speichern
BCØ5 ØF              RRCA
BCØ6 ØF              RRCA
BCØ7 ØF              RRCA
BCØ8 E6Ø7            AND     Ø7H
BCØA 47              LD      B,A       ;Operandenteil speichern
BCØB F1              POP     AF        ;Mnemonik-Nr. im A-Register
```

```
BCØC 185A            JR      PLA
BCØE 1A      E:      LD      A,(DE)    ;Adressumrechnung fuer
BCØF 13              INC     DE        ;Relativspruenge
BC1Ø 6F              LD      L,A
BC11 26ØØ            LD      H,ØØH
BC13 FE8Ø            CP      8ØH       ;Sprung vor o. zurueck ?
BC15 38Ø1            JRC     NCOR
BC17 25              DEC     H         ;rueckwaerts
BC18 19      NCOR:   ADD     HL,DE     ;abs. Adresse
BC19 7C              LD      A,H
BC1A CD28BC          CALL    PAH
BC1D 7D      PLH:    LD      A,L
BC1E 18Ø8            JR      PAH
BC2Ø 1A      NN:     LD      A,(DE)    ;16-Bit-Zahl
BC21 13              INC     DE
BC22 2E1D            LD      L,1DH     ;LOW(PLH)
BC24 E5              PUSH    HL
BC25 6F              LD      L,A
BC26 1A      N:      LD      A,(DE)    ;8-Bit-Zahl
BC27 13              INC     DE
BC28 CDØ3FØ  PAH:    CALL    OS
BC2B 1C              DB      1CH       ;Ausgabe von Reg. A in HEX
BC2C C9              RET
BC2D C63B    B:      ADD     3BH       ;Bitposition (Ø...7)
BC2F FE2B    PRIA:   CP      2BH       ;internen Zeichencode in
BC31 3812            JRC     MACRO     ;ASCII-Code umwandeln und
BC33 D6ØB            SUB     ØBH       ;ausgeben
BC35 FE5B            CP      5BH
BC37 38Ø7            JRC     PRA
BC39 D61A            SUB     1AH
BC3B CD4ØBC          CALL    PRA
BC3E 3E2Ø            LD      A,2ØH     ;SPACE
BC4Ø CDØ3FØ  PRA:    CALL    OS
BC43 24              DB      24H       ;ASCII-Zeichen ausgeben
BC44 C9              RET
BC45 E5      MACRO:  PUSH    HL
BC46 21D6BC          LD      HL,RET
BC49 E5              PUSH    HL
BC4A FE1D            CP      1DH       ;nur aus Tab. heraussuchen?
BC4C 381A            JRC     PLA
BC4E FE25            CP      25H       ;Macro-Programm anspringen?
BC5Ø 3Ø2A            JRNC    MA
BC52 FE1F            CP      1FH       ;Macro umrechnen und aus
BC54 38ØC            JRC     M2        ;Tabelle heraussuchen
BC56 2ØØ2            JRNZ    M1
BC58 CBCØ            SET     Ø,B
BC5A 3D      M1:     DEC     A
BC5B FE22            CP      22H
BC5D 38Ø3            JRC     M2
BC5F CB9Ø            RES     2,B
BC61 3D              DEC     A
BC62 87      M2:     ADD     A
BC63 87              ADD     A
BC64 87              ADD     A
BC65 D6DB            SUB     ØDBH
BC67 8Ø              ADD     B
BC68 CB49    PLA:    BIT     1,C       ;Zeichenkette aus Tabelle
BC6A 283E            JRZ     NM        ;heraussuchen
BC6C FE1A            CP      1AH       ;Register IX/IY ?
BC6E 2Ø27            JRNZ    NHL
```

```
BC7Ø 3E54            LD      A,54H     ;Buchstaben I
BC72 CD2FBC          CALL    PRIA      ;ausgeben
BC75 79      XY:     LD      A,C       ;X/Y ausgeben
BC76 E6Ø1            AND     Ø1H
BC78 C663            ADD     63H
BC7A 18B3            JR      PRIA
BC7C C6B3    MA:     ADD     ØB3H      ;Macro-UP-Ansprung
BC7E 6F              LD      L,A       ;B3H=LOW(MAC)-25H
BC7F 6E              LD      L,M       ;Adresse aus Tabelle
BC8Ø 78              LD      A,B
BC81 CB79            BIT     7,C       ;CB + DD/FD ?
BC83 E9              JP      (HL)      ;Ansprung
BC84 2Ø15    R:      JRNZ    M         ;8-Bit-Register
BC86 FD7D    RA:     LD      A,LY      ;Code-Byte
BC88 E6Ø7            AND     Ø7H       ;Bit Ø12
BC8A 47              LD      B,A
BC8B 3E2Ø            LD      A,2ØH
BC8D 18D3            JR      M2
BC8F C8      R1:     RZ                ;8-Bit-Register fuer
BC9Ø 3E37            LD      A,37H     ;CB-Befehle   ;Komma
BC92 CD2FBC          CALL    PRIA
BC95 18EF            JR      RA
BC97 FE2B    NHL:    CP      2BH       ;M ?
BC99 2ØØF            JRNZ    NM
BC9B CBF1    M:      SET     6,C       ;(IX+d)/(IY+d) aufgetreten
BC9D 3E39            LD      A,39H     ;(IX+d)/(IY+d) ausgeben
BC9F CB79            BIT     7,C       ;CB + DD/FD ?
BCA1 28Ø7            JRZ     NM
BCA3 1B              DEC     DE        ;Zeiger auf Ablage
BCA4 1B              DEC     DE
BCA5 CDAABC          CALL    NM
BCA8 13              INC     DE        ;Zeiger wieder zurueck
BCA9 C9              RET
BCAA 3C      NM:     INC     A
BCAB 215FBA          LD      HL,BMNE   ;Zeichentabelle
BCAE 23      SCH:    INC     HL        ;Zeichenkette suchen
BCAF CB7E            BIT     7,M
BCB1 28FB            JRZ     SCH
BCB3 3D              DEC     A         ;gefunden ?
BCB4 2ØF8            JRNZ    SCH
BCB6 7E      PRST:   LD      A,M       ;Zeichenkette ausgeben
BCB7 E67F            AND     7FH
BCB9 CD2FBC          CALL    PRIA
BCBC 23              INC     HL
BCBD CB7E            BIT     7,M       ;Ende Zeichenkette ?
BCBF 28F5            JRZ     PRST
BCC1 7D              LD      A,L
BCC2 E6FE            AND     ØFEH
BCC4 FECØ            CP      ØCØH      ;letztes Zeichen H- oder
BCC6 CØ              RNZ               ;L-Register ?
BCC7 CB49            BIT     1,C       ;DD/FD ?
BCC9 C8              RZ
BCCA CB71            BIT     6,C       ;(IX+d)/(IY+d) bereits
BCCC CØ              RNZ               ;aufgetreten ?
BCCD FD7D            LD      A,LY      ;Code-Byte
BCCF E6C7            AND     ØC7H
BCD1 FE46            CP      46H       ;LD r,(IX+d)/(IY+d) ?
BCD3 C8              RZ
BCD4 189F            JR      XY        ;X/Y anhaengen
BCD6 E1      RET:    POP     HL
```

```
BCD7 C9              RET
BCD8                 ;Macro-Ansprungtabelle
BCD8 26      MAC:    DB      26H       ;LOW(N)
BCD9 2Ø              DB      2ØH       ;LOW(NN)
BCDA ØE              DB      ØEH       ;LOW(E)
BCDB 2D              DB      2DH       ;LOW(B)
BCDC 84              DB      84H       ;LOW(R)
BCDD 8F              DB      8FH       ;LOW(R1)
BCDE                 ;Hauptprogramm
BCDE 7F7F            DA      7F7FH     ;Prolog
BCEØ 444953          DB      "DIS"
BCE3 Ø1              DB      Ø1H       ;Epilog
BCE4 3E1Ø    MAIN:   LD      A,1ØH
BCE6 32AØB7          LD      (CURS),A
BCE9 EB              EX      DE,HL
BCEA D5              PUSH    DE
BCEB CDDØBB          CALL    DIS       ;Mnemonik ausgeben
BCEE E1              POP     HL        ;alte Adresse
BCEF 3EØD            LD      A,ØDH     ;CR
BCF1 CD4ØBC          CALL    PRA
BCF4 CDØ3FØ          CALL    OS
BCF7 1A              DB      1AH       ;HL in HEX + SPACE ausgeben
BCF8 EB              EX      DE,HL
BCF9 ED52            SBC     HL,DE
BCFB 45              LD      B,L       ;Laenge des Befehls
BCFC EB              EX      DE,HL
BCFD 7E      PHEX:   LD      A,M       ;HEX-Code des Befehls
BCFE 23              INC     HL        ;ausgeben
BCFF CD28BC          CALL    PAH       ;A in HEX ausgeben
BDØ2 1ØF9            DJNZ    PHEX
BDØ4 CDØ3FØ          CALL    OS
BDØ7 2C              DB      2CH       ;CRLF
BDØ8 CDØ3FØ          CALL    OS
BDØB 16              DB      16H       ;Tastatur abfragen
BDØC FEØ3            CP      Ø3H       ;"Break" ?
BDØE C8              RZ
BDØF 18D3            JR      MAIN      ;weiter
BD11                 ;Ende
```

Bild 1 Assemblerlisting des Disassemblers

Literatur

/1/ Kieser, H.; Meder, M.: Mikroprozessortechnik. 3. Auflage, Berlin: VEB Verlag Technik 1985

/2/ Maudrich, G.: Weitere U-880-Befehle. Radio, Ferns., Elektron., Berlin 32 (1983) 11, S. 726

/3/ Thiel, W.: Befehlsstruktur des U880. Radio, Ferns., Elektron., Berlin 33 (1984) 7, S. 419–421

Entwurf von Steuerungssystemen für die flexible Fertigung

Rolf Hiersemann
Forschungszentrum des Werkzeugmaschinenbaues im VEB Werkzeugmaschinenkombinat „Fritz Heckert“
Karl-Marx-Stadt

1. Entwurfsphasen

Die Steuerungsentwicklung kann in die Phasen

1. Problemanalyse
2. Spezifikation
3. Entwurf und
4. Implementierung,

die sequentiell und teilweise iterativ ablaufen, unterteilt werden. Während in der **Problemanalyse** bisher vorrangig informale und verbale Darstellungen zur Anforderungsspezifikation (Pflichtenheft) wie

– technologische Prozeßmodelle,
– Steuerungs- und Bedienkonzeptionen und
– Auflistungen von Nebenbedingungen

angewandt wurden, ist in 7 /1/ ein Trend zu formalen Darstellungen zu erkennen.
Auf der Grundlage der Aussagen des Pflichtenheftes sind in der **Spezifikationsphase** die Grobstruktur des Steuerungssystems, der Einzelsteuerungen und deren funktionale Komplexe sowie die Art der Kommunikation zwischen den Einzelsteuerungen zu entwerfen. Prinzipiell sollte die Aufgabenaufteilung auf eine Leitebene und eine prozeßnahe Ebene (industrielle Steuerungen) erfolgen. Aufgabenspektrum und -umfang bestimmen dann Art, Anzahl und Ausbaustufe der einzusetzenden Steuerungen.
In der Phase **Entwurf** sind die Funktionen der einzelnen Steuerungskomponenten zu entwerfen. Dabei ist eine durchgängige hierarchische Gliederung anzustreben. Bei der Gliederung in Funktionsbausteine (FB) ist die Art und Weise der Wechselbziehungen zwischen diesen entscheidend. Anzustreben, jedoch praktisch kaum realisierbar, ist die reine Baumstruktur, bei der FB niederer Hierarchieebenen nur von einem FB der direkt übergeordneten Ebene benutzt werden können. Horizontale Wechselbeziehungen (auf gleicher Ebene) von FB sollten generell vermieden werden. Praktikabel sind Strukturen nach Bild 1, bei denen einige FB als Ressourcen übergeordneter Hierarchieebenen fungieren.
Prinzipiell ist eine Einteilung der FB in eine Koordinations- und eine Modulebene anzustreben. Die genaue Strukturierung und Unterteilung in verschiedene Ebenen ist das Ergebnis der Anwendung von Verfeinerungs- (top-down) und Vergröberungsstrategie (bottom-up) im Entwurfsprozeß: Praktikabel ist die Anwendung beider Strategien, das heißt, nach dem Entwerfen von Elementarfunktionen werden diese zu Abläufen (Teilmoduln) vergröbert, und ein grob entworfener Koordinator wird in Richtung Teilmoduln verfeinert. Dieses Vorgehen wird so oft wiederholt, bis nach mehreren Anläufen die letztendlich gültige Struktur und Funktion geschaffen wurde. Dabei sollten folgende Gesichtspunkte beachtet werden:

– Konstruktiv-technologische Zusammenhänge sind in entsprechenden Funktionsbausteinen widerzuspiegeln.
– Die Schnittstellen zwischen den FB sind zu minimieren.
– Zeitkritische Abläufe und parallele Prozesse sind in Abhängigkeit vom angewendeten Entwurfsverfahren auf einzelne FB abzubilden.

Besondere Beachtung ist in der Entwurfsphase dem frühzeitigen Test einzelner Funktionen zu widmen, um funktionale und logische Fehler zu orten und zu beheben.
Die **Implementierung** stellt die Umsetzung der entworfenen Funktionen in die der Steuerung verständliche Sprache und deren Inbetriebnahme dar. In dieser Phase sind Wiederholungen der Entwurfsphasen 2 bis 4 oft unumgänglich, um eine optimale Steuerungsfunktion zu erreichen.

2. Entwurf der Kommunikationssteuerung

Für den Zusammenschluß von Einzelsteuerungen zu Verbundsystemen existiert eine Reihe von Möglichkeiten. Prinzipiell ist die Verbindungsart in serielle und parallele Kopplung und die Speicherkopplung (Nutzung gemeinsamer externer Speicher oder Arbeitsspeicherbereiche) unterteilbar, wobei folgende physische Strukturen realisierbar sind (vgl. Bild 2):

- parallele Kopplung
– Punkt-zu-Punkt-(Kanal-Kanal-) Kopplung (direkt, Adapter)
– parallele Busse

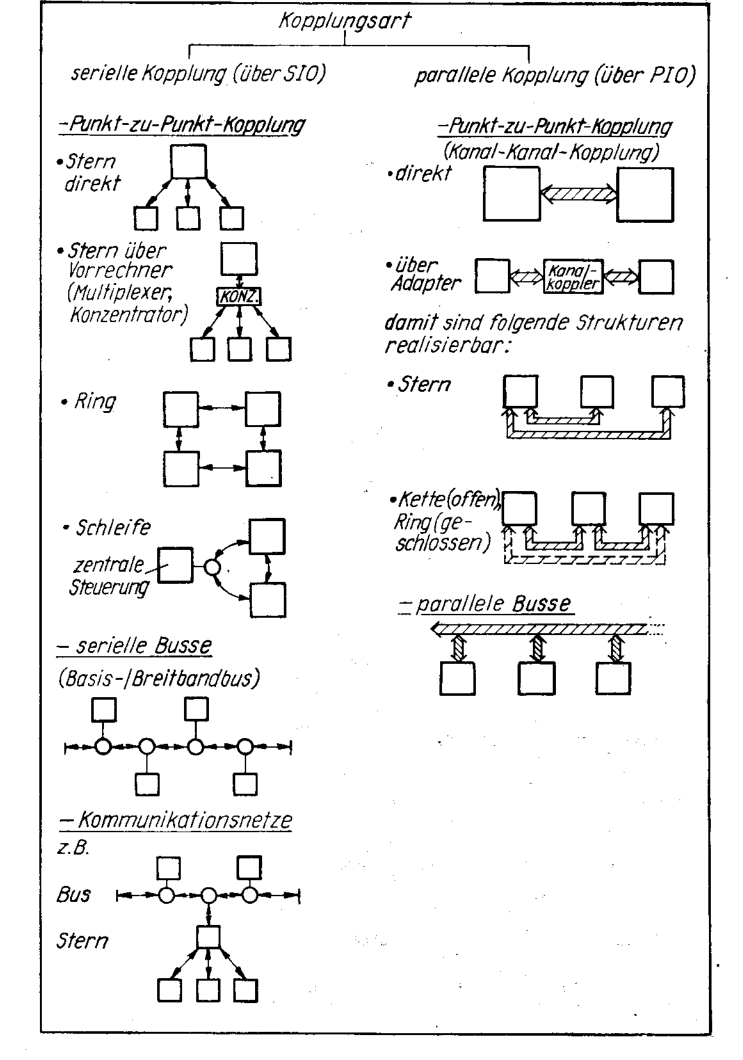


Bild 2
Physische Strukturen von Kommunikationssteuerungen bei serieller und paralleler Kopplung

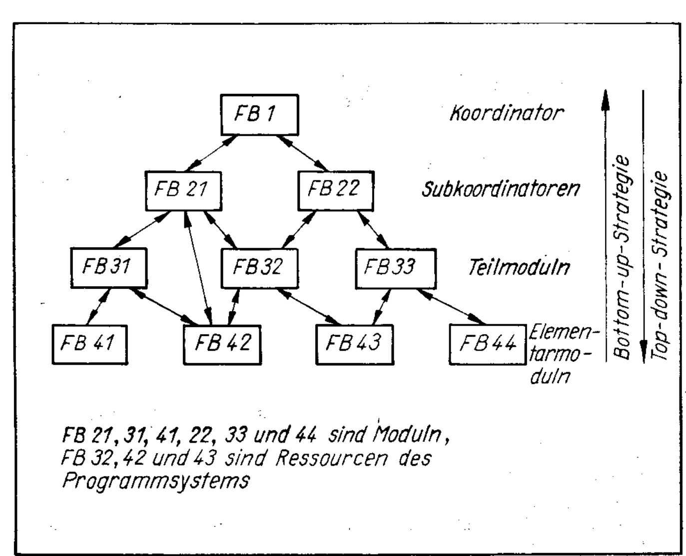


Bild 1
Zusammenhang Modularisierungsstrategien und Hierarchie, unterteilt in Funktionsbausteine (FB).
FB 21, 31, 41, 22, 23 und 44 sind Moduln, FB 32, 42 und 43 sind Ressourcen des Programmsystem.

- serielle Kopplung
  - Punkt-zu-Punkt-Kopplung (direkt, Vorrechner)
  - serielle Busse
  - Kommunikationsnetzwerke.

Die Entscheidung für eine spezielle Kopplungsart ist in erster Linie abhängig von der Verfügbarkeit entsprechender Gerätetechnik. In größeren Systemen werden oftmals mehrere Kopplungsarten zugleich angewandt (Bild 3). Die in der Spezifikationsphase festzulegende physische Struktur des Steuerungsverbundes sollte so erfolgen, daß eine abgestufte Funktionsfähigkeit des Gesamtsystems bei Ausfall von einzelnen Komponenten gewährleistet (z. B. durch Redundanzen in der gerätetechnischen Auslegung erreichbar) und der geforderte Informationsfluß pro Zeiteinheit verlustfrei garantiert werden.
Damit erlangen die logische Auslegung der Kommunikationssteuerung, die in

- Konkurrenzprinzip
- Unterordnungsprinzip und
- Abfrageprinzip

unterteilt werden kann, und die Festlegung der zu übertragenden Datenformate und -inhalte (Nettotext) größere Bedeutung. In den flexiblen Maschinensystemen (FMS) des Werkzeugmaschinenbaus wurde in der Steuerungsebene die parallele (Kanal-Kanal-)Kopplung der Einzelsteuerungen gewählt, um bei kleinen und mittleren Entfernungen (10 . . . 100 Meter) große Übertragungskapazitäten zu gewährleisten und bei Ausfall aller übergeordneter Gerätetechnik (Prozeßleit-, Steuer- und Komplexrechner) eine Funktionsfähigkeit der Maschinensteuerungen (CNC) im Zusammenwirken mit dem Transportmittel STR (schienengebundener Transportroboter) über die zentrale Anlagensteuerung (Speicherprogrammierbare Steuerung PC 600) zu garantieren. Für die serielle Kopplung der übergeordneten Rechner wurde die Sternstruktur (direkt und Nutzung von Vorrechnern – Konzentrator) bei Anwendung einer einheitlichen Standardprozedur (BAC – Binary Asynchronous Communication – asynchrone Konkurrenzprozedur) gewählt. Damit ist die Möglichkeit zur Durchsetzung einer abgestuften Inbetriebnahme- und Teststrategie bei Nutzung eines einheitlichen Testhilfsmittels in Form des Prozedur-Test- und Simulationsgerätes PROTESI auf Bürocomputerbasis gegeben. Mit dem Einsatz speicherprogrammierbarer Steuerungen in den flexiblen Maschinensystemen mußte auch deren Funktionsumfang um die Realisierung einer seriellen Schnittstelle zum Rechnerverbund erweitert werden. Gerätetechnisch bedeutet das bei den eingesetzten Steuerungen vom Typ PC 600 das Stecken der Kartenbaugruppe DNC auf den Rechnerbus. Auf dieser Kartenbaugruppe ist auch die Prozedur BAC resident verfügbar. Entscheidend für den Steuerungswert ist somit die funktionale Einbindung der Prozedur ins Anwenderprogramm.
Bedingt durch das große Einsatzspektrum speicherprogrammierbarer Steuerungen, das in flexiblen Maschinensystemen von Maschinen- über Anlagen- bis hin zu Transportgerätesteuerungen reicht, wurde in der Steuerung PC 600 die Einbindung der Proze-

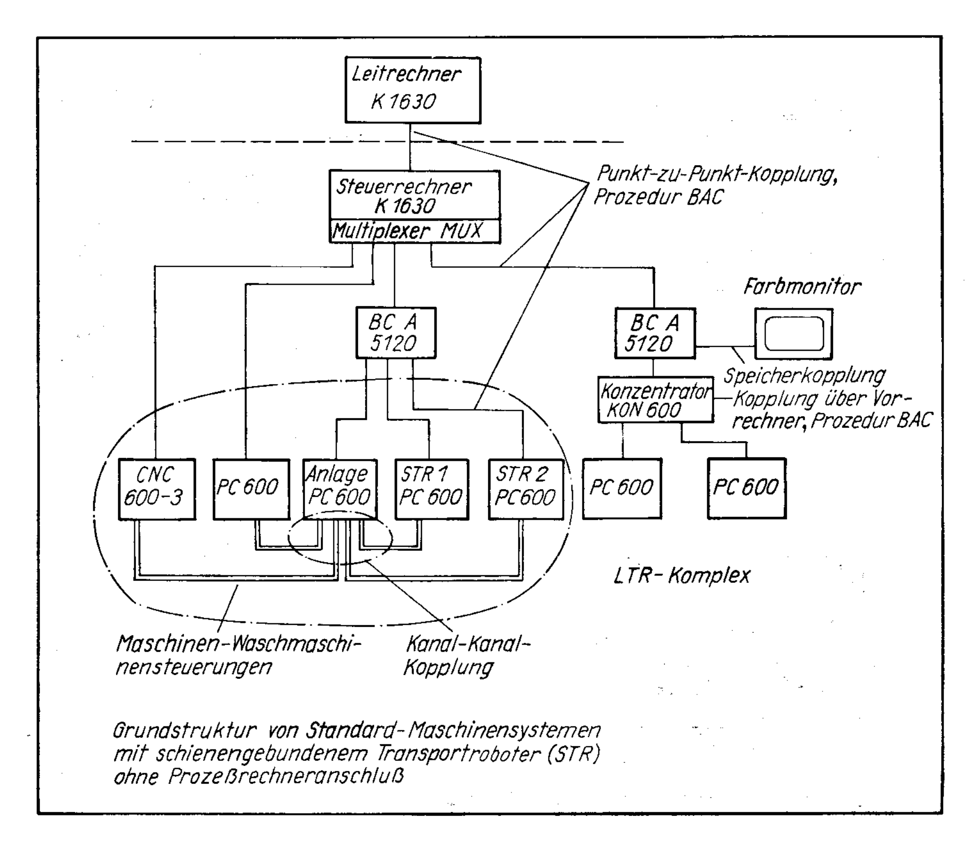


Bild 3
Ausschnitt aus der Steuerungsstruktur des flexiblen Maschinensystems FMS 1000

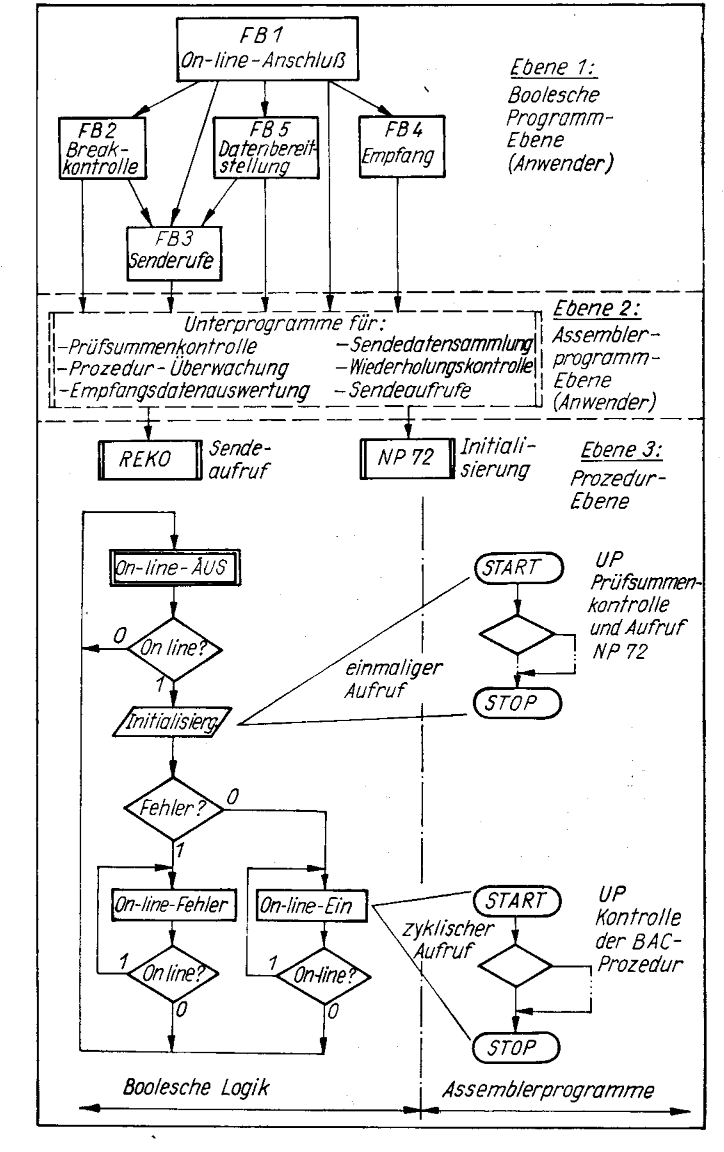


Bild 4
Einbindung der seriellen Datenübertragungsprozedur BAC ins Anwenderprogramm der Speicherprogrammierbaren Steuerung PC 600.
a) 3 Ebenen der Programmrealisierung
b) Zusammenhang zwischen Boolescher Logik (entworfen als PAG) und Maschinencodeprogramm (entworfen als PAP)

Dipl.-Ing. Rolf Hiersemann (32) ist gelernter Elektromonteur und studierte von 1976 bis 1981 an der Sektion Automatisierungstechnik der Technischen Hochschule Karl-Marx-Stadt im Wissenschaftsbereich Technische Kybernetik. Anschließend wurde er Entwicklungsingenieur im Bereich Mikroelektronik des Forschungszentrums des Werkzeugmaschinenbaues Karl-Marx-Stadt (FZW) und arbeitete auf dem Gebiet der Programmentwicklung für Industrieroboter- (IRS 2000, IRS 610) und Transportgerätesteuerungen für flexible Fertigungssysteme. Er leitete die Systemprogrammierung des Standardmaschinensystems FMS 800. Gegenwärtig arbeitet er an der Entwicklung einer Diagnoseeinrichtung für Taktstraßensteuerungen. Sein Spezialgebiet ist der systematische Entwurf industrieller Steuerungen, hierzu geht er auch einer außerplanmäßigen Aspirantur an der Technischen Universität Karl-Marx-Stadt nach.

dur BAC vollständig über das Anwenderprogramm realisiert und das Betriebssystem nicht verändert. Damit kann der Entwerfer alle Möglichkeiten der Prozedur ausschöpfen, ist aber auch zur Programmierung in zwei Sprachniveaus (Boolesche Logik und Assemblerprogrammierung des Prozessors U 880) gezwungen. Um den Realisierungsaufwand für die Einbindung der BAC-Prozedur ins Anwenderprogramm zu reduzieren, wurde in /3/ eine Entwurfsmethode vorgeschlagen, die darin besteht, als Beschreibungsmittel den Programmablaufgraph (für Boolesche Programmteile) und den Programmablaufplan (für Assemblerprogrammentwicklung) einzusetzen. Die Kopplung der einzelnen Funktionsbausteine (Boolesche Logik) und Unterprogramme erfolgt über gemeinsame Arbeitsspeicherbereiche (Z-Variable). Entsprechend Bild 4a werden von der Ebene 1 aus die Unterprogramme der Ebene 2 aktiviert, von denen aus der Zugriff auf die BAC-Prozedur möglich ist. Durch den Aufruf von instabilen Zuständen aus (Bild 4b, Zustand Initial.) wird ein einmaliger Aufruf von Unterprogrammen realisiert; zyklisch aufzurufende Programme sind an stabile Zustände (Zustände mit Eigenschleife) zu binden.

Aufbauend auf den Funktionsentwürfen wird über bewährte Umsetzungsverfahren aus den Programmablaufgraphen (PAG) die Boolesche Logik entwickelt. Dazu ist jedem Zustand eine Speichervariable zuzuordnen und die Notation in Form von setzenden Flip-Flops durchzuführen:

Zustand = Setzterm + Halten./Rücksetzterm

$$Z_i = \bigvee_{j\neq i} z_j * h_{ji} + Z_i * \bigwedge_{k\neq i,0} /Z_k \qquad (1)$$

j ... Index der Vorzustände zum Zustand Z_i

k ... Index der Folgezustände zum Zustand Z_i

h_{ji} ... logische Bedingung für den Zustandswechsel $Z_j \rightarrow Z_i$

Instabile Zustände sind als kombinatorischer Ausdruck ohne Speicherwirkung zu notieren:

$$Z_i = \bigvee_{j\neq i} Z_j * h_{ji} \qquad (2)$$

Die Programmnotation erfolgt satzweise, der Zugriff auf das Maschinencodeniveau wird durch Überspringen einzelner Sätze realisiert. Soll z. B. vom Zustand „On-line Ein" im Bild 4b das Unterprogramm „BAC-Kontrolle" aktiviert werden, ist folgende Programmnotation zu realisieren :

Nxx1: $Z_{\text{on-line EIN}}$ = ...
Zustandsgleichung

Nxx2: Nxx4 * /$Z_{\text{on-line EIN}}$,
Übersprung des UP

Nxx3: B = . . .,
UP „BAC-Kontrolle"

Nxx4: Z . . . =,
nächster Satz = Sprungziel von Nxx2

3. Angepaßte Programmsysteme

Eine generelle Wiederholung eines Maschinensystems ohne Veränderung ist so gut wie ausgeschlossen. Vielmehr müssen bei Projektierung und Entwurf neuer Maschinensysteme einzelne Bausteine bekannter Systeme so zusammengefügt werden, daß bei Minimierung von Entwicklungszeit, -risiko und -kosten optimale Lösungen entstehen. Die Anwendung des Baukastenprinzips betrifft dabei nicht nur den klassischen Sektor des Maschinenbaus, sondern ist vor allem in den Steuerungen der unteren Hierarchieebenen bei der Konfiguration von Hard- und Software zu beachten.

Ein hoher Anteil von Maschinensystemen (Bild 5) weist eine Grundkonfiguration, bestehend aus mehreren Bearbeitungszentren (4...10 BAZ), einer Wasch- und Kühlstation (WAKUE), 4 Spannplätzen und mehreren Speicherplätzen, verbunden durch ein oder zwei schienengebundene Transportroboter (STR), auf, die den Einsatz wiederverwendbarer Software-Moduln ermöglicht. Für dieses Einsatzgebiet wurde ein Programmsystem /4/ entwickelt, das für speicherprogrammierbare Steuerungen PC 600 als Anlagen-

Bild 5 Anordnungsschema (Layout) des Maschinensystems FMS 1000

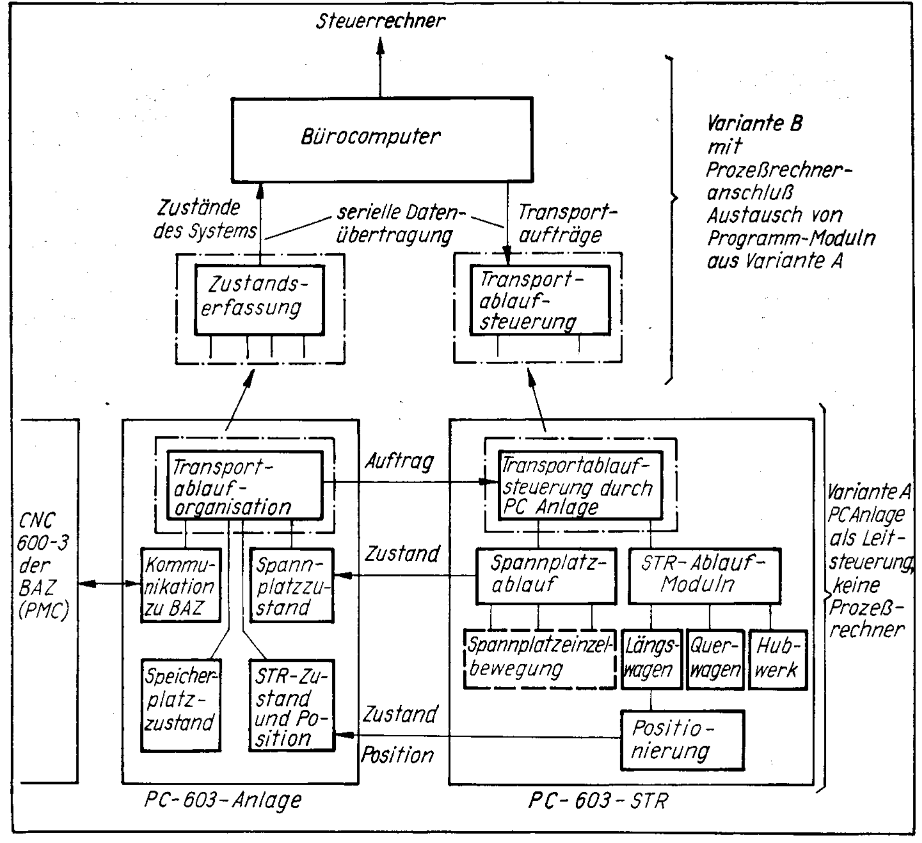


Bild 6 Funktionsprobleme des Programmsystems TUL-Komplex STR

bzw. STR-Steuerungen anwendbar ist. Das Prinzip dieses Programmsystems besteht darin, den Anwender von der Assemblerprogrammierung zu befreien, indem nur maschinensystemspezifische Generierdaten festzulegen sind (vgl. Bild 4a), und für die Programmierung im Niveau der Booleschen Logik einen Grundsatz vorzugeben, der entsprechend der Maschinensystemkonfiguration abzuwandeln ist.

Für die serielle Datenübertragung wurden Datenformate und -inhalte (Nettotext) festgelegt, die auch für den Komplex LTR angewendet wurden und Erweiterungen zulassen.

Die Steuerungsgrobstruktur (siehe Bild 3) wird durch die Zuordnung der Funktionskomplexe
- Kommunikation mit BAZ
- Transportablauforganisation oder Zustandserfassung (als Information für Bürocomputer BCA)
- Speicherplatzzustandserfassung

→ zur „Anlagen"-Steuerung (PC-Anlage) und
- Transportablaufsteuerung (Aufträge, ausgehend von PC-Anlage oder Bürocomputer)
- Einzelfunktionen STR
- Ablauf- und Einzelfunktionen Spannplätze

→ zur Gerätesteuerung (PC-STR) untersetzt. Damit sind gleichbleibende Funktionskomplexe in der PC-STR lokalisiert. Diese PC 600 kann somit nahezu unverändert in verschiedenen Maschinensystemen eingesetzt werden. Der stark anlagenspezifische Anteil ist in der PC-Anlage lokalisiert, die entsprechend zu modifizieren ist. Wird das Maschinensystem ohne Prozeßrechner realisiert, übernimmt die Anlagensteuerung die Bildung der Transportaufträge und leitet diese über Kanal-Kanal-Kopplung zur PC-STR weiter, die diese Aufträge ausführt (Bild 6, Version A). Bei übergeordnetem Komplexrechner (Bürocomputer - BCA) erhält jede Steuerung einen seriellen Anschluß. Über diesen werden von der PC-Anlage an den BCA die Zustandsdaten des Maschinensystems übermittelt, aus denen im BCA die Aufträge gebildet und an die PC-STR gesendet werden (Bild 6, Version B). Die Anwendung dieses Programmsystems ist auf den Anschluß von

- max. 12 Bearbeitungszentren (incl. WAKUE)
- max. 7 Spannplätzen
- max. 24 wahllos angeordneten Speicherplätzen
- max. 32 lückenlos angeordneten Speicherplätzen und
- 1 oder 2 schienengebundenen Transportrobotern

ausgelegt und an die Einhaltung von Richtlinien zur Konfigurierung der Hardware gebunden, wodurch die Entwicklungszeiten sinken, da nachnutzbare Lösungen vorliegen. Der Gewinn an Entwicklungszeit wird pro Steuerung auf etwa 3 Monate an ingenieurtechnischer Arbeit eingeschätzt.

4. Ausblick

Beim Entwurf von Steuerungssystemen für die flexible Fertigung sind vorrangig Lösungen zu schaffen, die bei hohem Nachnutzungsgrad geringe Entwicklungszeiten, -kosten und -risiken ermöglichen. Im Entwurfsprozeß sind zunehmend formale Darstellungsmittel auf der Basis von Netzen einzusetzen, um durch Analyse- und Simulationsmöglichkeiten frühzeitig Fehler orten und beheben zu können. Auch die Nutzung von dem Entwurfsverfahren angepaßten Programmiertechnologien reduziert den Aufwand gegenüber Methoden zur manuellen Umsetzung der Funktionen in das Sprachniveau der eingesetzten Steuerung (wie im Beitrag beschrieben) erheblich.

Für Wiederhollösungen von Fertigungssystemen ohne Prozeßrechnerebene vom Typ FMS 800 wurden basierend auf dem in /1/ erläuterten Vorgehen für die Koordinationsebene Modelle in Form von Prädikat-Transitions-Netzen erstellt. Mit diesen Modellen wurden Simulationen zur Gewinnung der technologischen Grenzen und der Vorrangkriterien für Transportaufträge durchgeführt. Dadurch konnten Abläufe, die gegenüber den derzeit realisierten und in /1/ dargestellten festen Transportzyklen (Transport Palette:

Spannplatz → BAZ, WAKUE → Spannplatz, BAZ → WAKUE) aktive Speicherplätze zum Zwischenlagern und Transporte BAZ → BAZ zur Realisierung von Bearbeitungsfolgen über mehrere Maschinen hinweg einbeziehen, entworfen werden. Damit werden derartige Fertigungssysteme einen Funktionsumfang erlangen, der Systemen mit Prozeßrechnerebene nahekommt.

Literatur

/1/ Fensch, S.: Anforderungsspezifikation und Modellbildung auf der Basis von Netzen. Mikroprozessortechnik Berlin 1 (1987) 10

/2/ Hartleib, H.; Zumpe, J.: Rechner- und Steuerungskonzepte für flexible Fertigungssysteme. IKM 1986 Karl-Marx-Stadt, Tagungsunterlagen, Vortrag 48

/3/ Nötzold, L.; Hiersemann, R.: On-line-Schnittstellenbeschreibung PC 600. Forschungszentrum des Werkzeugmaschinenbaues Karl-Marx-Stadt, 1984

/4/ Hiersemann, R.; Wolfram, G.: Anwenderrichtlinie des Programmsystems für Steuerungen des TUL-Komplexes STR. Nachnutzungsmaterial des VEB Werkzeugmaschinenkombinat „FRITZ HECKERT" – Stammbetrieb Karl-Marx-Stadt, 1986

✉ KONTAKT ☏

Forschungszentrum des Werkzeugmaschinenbaues Karl-Marx-Stadt, Abt. 219,
Karl-Marx-Allee 4, Karl-Marx-Stadt, 9010;
Tel. 599414

Autostart für A 5120

1. Vorbemerkung

In der Version 5.1 des Betriebssystems SCPX PC-1715 ist eine Autostart-Routine vorhanden, die einen automatischen Programmaufruf nach Kalt- und Warmstart durchführt. In /1/ wurde beschrieben, wie diese Funktion in CP/M und kompatiblen Systemen realisiert wird. Damit läßt sich auch für SCPX 1520 (A 5120, A 5130) ein Autostart erreichen (vgl. /2/). Die folgenden Erläuterungen beschreiben eine gegenüber /2/ vereinfachte Vorgehensweise für die Version SCP 1520 V1.5, lassen sich aber sinngemäß auf SCP 1520 V1.3 übertragen. Änderungen im Betriebssystem sollten aber nur von erfahrenen Programmierern auf Disketten vorgenommen werden, von denen vorher eine Sicherheitskopie angelegt wurde!

2. Grundlagen

Der Eingabepuffer des CCP enthält eine Eintragung über die Länge des Puffers (7FH), die Länge der aktuellen Eingabezeile (03H) sowie den eingegebenen Befehl (DIR) und ein abschließendes 00H, zum Beispiel:

```
7F  03  44  49  52  00
 .   .   D   I   R   .
```

Im Normalfall ist die Länge der Eingabezeile auf 00H gesetzt, woran der CCP erkennt, daß kein Befehl abzuarbeiten ist. Das Betriebssystem läßt sich so manipulieren, daß bei Kalt- und Warmstart mit dem CCP stets ein Kommando von der Diskette gelesen wird.

3. Realisierung

Beim Kopieren des Betriebssystems von Diskette zu Diskette lädt das Programm SYSG zunächst alle Komponenten in den TPA. Änderungen müssen vor dem Rückschreiben auf die Diskette mit dem Debugger DU vorgenommen werden. Dazu wird aufgerufen:

DUSYSG.COM

SYSG wird gestartet und dabei ein Haltepunkt auf die Adresse 2CBH gesetzt. Entsprechend der Bedienerweisung wird das System vom Laufwerk A geladen:

```
g, 2cb
SOURCE DRIVE NAME...:a
SOURCE ON A THEN TYPE ⟨ENTER⟩_
#
```

Nach dem Laden wird das Programm am Haltepunkt unterbrochen. Mit einer Dump-Anweisung wird kontrolliert, ob ab Adresse 9EDH die Meldung ROBOTRON LOADER zu finden ist:

d9e0

Der uns interessierende Bereich befindet sich ab Adresse A00H (bei SCP 1520 V1.3 ab Adresse 980H).

Hinter zwei Sprungbefehlen beginnt der Eingabepuffer:

A00: C3 6D CB C3 69 CB 7F 00 00 00 00

Auf A06H steht die maximale Länge des Puffers. Unmittelbar dahinter werden mit

sa07

die Länge des Autostart-Befehls und der Befehl selbst eingeschrieben:

03 44 49 52

Zum Abschluß wird mit

g

das Programm SYSG bis zum Schluß abgearbeitet. Dabei ist es gleich, ob das geänderte Betriebssystem auf die Quelldiskette oder ein anderes Laufwerk zurückgeschrieben wird.

DESTINATION DRIVE...: a

...

4. Anwendung

Autostart-Funktionen beschränken sich selbstverständlich nicht auf residente Kommandos. Alle von der Tastatur möglichen Eingaben lassen sich für den Autostart verwenden. Für benutzerfreundliche Programmstarts eignen sich vor allem parame-

(Fortsetzung auf Seite 349)

Kinder im Informationszeitalter

Im Vortragsprogramm dieses Kongresses, der vom 19. bis 23. Mai 1987 in Sofia stattfand, befanden sich 60 Beiträge aus 18 Ländern. Die sachliche Einteilung erfolgte lt. Programmkomitee in 3 Gebiete:

① Auf dem Weg zu einem neuen Verständnis der Bildung

② Informatik in und außerhalb der Schule

③ Erziehungs-/Lerntheorie und wissenschaftsorientierte Systeme.

Dem Gebiet ① wurden 20 Vorträge zugeordnet, dem Gebiet ② 22, und 18 Vorträge fanden zum Gebiet ③ statt. Trotz dieser (fast) Gleichverteilung war das Gebiet ③ offenbar aus organisatorischen Gründen mit nur 13 dem Thema inhaltlich echt verbundenen Vorträgen deutlich schwächer vertreten. Dazu kommt, daß der dargebotene Inhalt öfter über die konzeptionelle Phase – „man müßte", „man könnte" – nicht hinauskam. Bezüglich der Lerntheorie unter dem Aspekt des Einsatzes neuer technischer Mittel (Computer) klafft offenbar eine weltweite Lücke. Sie kann m. E. nur durch Zusammengehen von Erziehungswissenschaftlern und Informatikern geschlossen werden. Der Alleingang des sehr sympathischen und im persönlichen Auftreten recht bescheidenen *Seymour Papert* (vom MIT in USA, Kreation von LOGO) beweist das. In seinen überhöhten Ansprüchen tritt er inhaltlich wissenschaftlich, bez. der Erziehungstheorie, nur noch in der Form bescheiden auf.
Allgemein gilt, daß zeigte auch diese Tagung, daß die große Mehrheit der Wissenschaftler das Eindringen des Computers in die Pädagogik/Didaktik akzeptieren muß. Es fehlt jedoch noch immer die erziehungstheoretische Basis zum Erkennen der genauen Grenzen eines exakt umrissenen Gebietes der Einsatzziele, Methoden und Möglichkeiten.

Computer in General Education

Vom 16. bis 18. Mai 1987 fand in zeitlicher Nähe des Kongresses eine Konferenz *Informatics and Teaching Mathematics* statt, die von der IFIP Technical Committee 3 „Education" der Working Group 3.1 organisiert wurde. Es nahmen etwa 50 Wissenschaftler aus 18 Ländern teil. Die weitesten Reisen hatten 9 Teilnehmer aus Kanada, Australien, Argentinien und den USA. Stärker als andere Länder waren England mit 10 Personen und der Block der skandinavischen Länder mit 8 Personen vertreten. Von den sozialistischen Ländern – neben Bulgarien – waren noch Ungarn, die UdSSR und die DDR mit je 1 Wissenschaftler vertreten. Diese Verteilung macht etwa deutlich, welche Beachtung dem Tagungsgegenstand von den einzelnen Staaten gewidmet wird.
Die Gliederung und thematische Zuordnung der 17 Vorträge läßt etwa folgende Einteilung zu: Sprachen, Hilfsmittel, Maßnahmen zur Förderung, Länderberichte und das Verhältnis von Informatik zur Mathematik in der schulischen Allgemeinbildung.
In der Gruppe Sprachen wurden prozedurale Sprachen in ihrem Einfluß auf die Mathematikausbildung einerseits und der Einfluß der Mathematik auf Sprachen andererseits behandelt.
Bei den Hilfsmitteln, welche die Informatik der Mathematikausbildung geben kann, fanden Programmpakete zur Graphik, zur Behandlung von numerisch-analytischen Problemen und zur Simulation und Modellierung besondere Beachtung. Die vorgestellten Maßnahmen zur Förderung beinhalteten Lerntheorie, Formen der Lehrausbildung in Mathematik, Lehrprogrammfragen unter Beeinflussung durch IT, Informatik in der Schule und neue Entwicklungen der Informatik.

Prof. P. Kenderov (Bulg. Akademie d. Wissenschaften) sprach über einen von Bulgarien organisierten internationalen Wettstreit in Informatik für Schüler, ähnlich unserer bekannten Mathematik-Olympiade. Zur Verfügung standen Apple- bzw. IBM-kompatible PC bulgarischer Produktion.
Das Verhältnis der Informatik zur mathematischen Ausbildung wurde von *C. Hoyles* (England) und *A. Ralston* (USA) bez. der Klassenstufen getrennt betrachtet.
Ferner wurden einige Fragen und Probleme dieses Verhältnisses in vier kleineren Diskussionsgruppen recht ausführlich, aber nicht umfassend behandelt. Die Resultate wurden dem Plenum vorgestellt.
Es herrscht im wesentlichen Übereinstimmung darüber, daß die Informatik auch auf die Schulmathematik sowohl in der Stoffauswahl als auch in der Zuordnung zu den Alters- bzw. Klassenstufen Einfluß nehmen wird und teilweise schon nimmt. Das muß von der Ausbildung für Mathematiklehrer reflektiert werden.
Es wurde eine Reihe von Länderberichten gegeben (USA, Österreich, England, Schweden, DDR). Dabei stellte sich heraus, daß unser Standpunkt zwar wissenschaftlich, gesellschaftswissenschaftlich und pädagogisch mit am besten begründet werden konnte, die Realisierung jedoch – auch gegenüber anderen sozialistischen Ländern – einen Rückstand aufweist.

Ausstellung

In einer recht umfangreich angelegten Ausstellung mit ausländischer Beteiligung zeigte im wesentlichen Bulgarien selbst seine Leistungsfähigkeit im Computerbau nebst Zubehör. Durch den Erwerb von Lizenzen von Apple und IBM werden auch für den Unterricht und das Bildungswesen sehr gut geeignete Geräte in ausreichender Zahl hergestellt. Bulgarien arbeitet auf dem Gebiet „Polytechnik-Roboter" mit Fischer-Elektronik (BRD) und einer japanischen Gruppe „Medicom Systems" zusammen. Diese Vereinigung produziert das Schulrobotersystem ROBCO, welches aufbaufähig und sehr leistungsstark ist. Es wird zum Verkauf angeboten.

Prof. Dr. I. O. Kerner

FORTH-Erfahrungsaustausch

Am 13. 6. 1987 trafen sich in der Berufsschule „Friedrich Engels" in Döbeln erstmalig Computeramateure zu einem Erfahrungsaustausch über die Arbeit mit der Programmiersprache FORTH auf Heim- und Kleincomputern. Auf der Tagesordnung standen die Vorstellung von FORTH-Systemen, die Demonstration von in FORTH geschriebener Anwendersoftware und die Beratung über einen effektiven Austausch von Erfahrungen, Literatur und selbsterstellter Software. Eine Diskussion der unbestreitbaren Vorteile von FORTH gegenüber der sonst auf Heimcomputer meist verwendeten Sprache BASIC war im Kreise dieser FORTH-Freunde nicht nötig. Offenbar haben aber bisher nicht nur viele Amateure, sondern auch viele Berufsprogrammierer die enorme Leistungsfähigkeit dieser Sprache, die sich zum Beispiel in größtmöglicher Transparenz und Geräteunabhängigkeit sowie in geringem Speicherbedarf und großer Abarbeitungsgeschwindigkeit darstellt, nicht erkannt. Ein Grund dafür mag die bisher unzureichende Anzahl von Publikationen zu dieser eleganten und leistungsstarken Dialogprogrammiersprache sein.
Es wurden FORTH-Kerne für die Computertypen KC85/3, KC85/1, Z1013, ZX81 und ZX Spectrum sowie CP/M-kompatibler Computer vorgestellt. Es existiert eine Version, die auf den Geräten KC85/3, KC85/1 und Z1013 einsetzbar ist. Da dafür offenbar ein großer Bedarf bei den Anwendern und Besitzern dieser Computer besteht, wäre es wünschenswert, wenn sich ein Vertreiber finden würde.
Weiterhin wurde eingeschätzt, daß ein Einchipmikrocomputer vom Typ U883, der anstelle eines BASIC-Interpreters einen FORTH-Kern im Masken-ROM beinhalten würde, nicht nur im Amateurbereich, sondern vor allem in der Industrie vielfältigste, sehr ökonomische Einsatzmöglichkeiten finden würde. Bei der Vorstellung von in FORTH geschriebener Anwendersoftware wurden zum Beispiel folgende Problemlösungen gezeigt:

- FORTH-Assembler für U880 und U882
- Schnelle Gleitkommaarithmetik
- Schnelle Grafik
- Schnelle ein- und mehrdimensionale Fouriertransformation
- Decompiler für Secondary- und Primitiv-Worte
- Ansteuerung von Peripherie (z. B. Plotter, EPROM-Brenner)
- Sprachausgabe.

Es wurde die Hoffnung geäußert, daß in nächster Zukunft die Zahl der an FORTH interessierten Computeramateure, resultierend aus der Leistungsfähigkeit dieser Programmiersprache, stark zunehmen wird.

T. Noßke, H.-J. Gatsche

✉ KONTAKT ☎

VEB Döbelner Beschläge
und Metallwaren,
Berufsschule „Friedrich Engels"
Kennwort FORTH, Döbeln, 7300

Tonbandanschluß für Mikrorechner auf der Basis des K 1520

An der Technischen Universität Dresden, Sektion Informationstechnik wurde ein universell einsetzbares Interface zum Anschluß handelsüblicher Kassettenmagnetbandgeräte an Mikrorechner auf der Basis des K 1520 geschaffen. Neben der Ankopplung von Magnetbandgeräten bietet es durch Bereitstelltung zweier PIO- sowie dreier CTC-Kanäle die Möglichkeit der Ansteuerung weiterer peripherer Einheiten. Es wurden Software-Module entwickelt, die das Speichern von Dateien auf Audiokassetten, innerhalb der Betriebssysteme CP/M und MEOS, im KC 85/1-kompatiblen Datenformat ermöglichen. Bei Einsatz dieses Interfaces werden folgende Effekte erzielt:
- Entlastung der zur Verfügung stehenden Disketten
- Möglichkeiten des Dateitransfers zwischen ansonsten inkompatiblen Betriebssystemen
- Möglichkeit der dezentralen Datenerfassung mit dem Kleincomputer KC 85/1
- Ersatz von teueren EPROMs durch Magnetbandkassetten bei der Programmübertragung zum Test innerhalb einfacher Rechnerkonfigurationen.

Technische Universität Dresden, Sekton Informationstechnik, Bereich Kommunikations- und Computertechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4635121

Herzmann

(Fortsetzung von S. 347)

trierte Befehle wie Kommandostapel oder REDABAS-Programme, z. B.:
SUBM STAPEL oder
REDABAS B: PROGNAME
Programme, die mit Warmstart enden (TURBO-PASCAL-Programme oder SYSTEM-Rückkehr aus BASIC-Programmen) starten sich selbst bei Programmende von neuem, was einen gewissen Schutz gegen unerlaubte Manipulationen (Löschkommando) bietet.

M. Lennartz

Literatur

/1/ Zaks, R.: CP/M-Handbuch mit MP/M. Sybex-Verlag Düsseldorf 1981, S. 203 ff.

/2/ Systemunterlagendokumentation MOS K 1520. Anwenderdokumentation. Anleitung für den Systemprogrammierer. VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt 1984, S. 5f.

PLZ-C-Sprachübersetzer

Der PLZ-C-Sprachübersetzer dient der Portierung von PLZ-SYS-Quell-Programmen in äquivalente Programme der Sprache C. Er wurde entwickelt, um unter dem Betriebssytem UDOS implementierte PLZ-Programme auf einem UNIX-kompatiblen Betriebssystem nutzbar zu machen. Bis auf wenige Ausnahmen können mit diesem Übersetzer alle Anweisungen von PLZ in semantisch äquivalente Anweisungen der Sprache C überführt werden (Einschränkungen gibt es bei der Initialisierung von Feldern und Strukturen). Eine vollständige Syntaxanalyse wird nicht vorgenommen, da davon ausgegangen wird, daß die zu portierenden Programme bereits syntaktisch fehlerfrei sind. Zusätzlich wurde für die Ein-/Ausgabe ein Satz von Routinen entwickelt, die die entsprechenden PLZ-Ein-/Ausgabe-Prozeduren (aus PLZ IO und PLZ.FIO) für das UNIX-kompatible Betriebssystem realisieren.

VEB RFT Nachrichtenelektronik Leipzig „Albert Norden", Postfach 15, Leipzig, 7027

Jäger

Dokumentation zu MS-DOS und dBase III

Die Dokumentation entstand zum Zweck der Durchführung von Seminaren zu MS-DOS und dBase III, wobei gründliche Kenntnisse von dBase II unter CP/M-80 vorausgesetzt werden. Sie soll und kann ein Lehrbuch, die Systemunterlagen oder den Lehrgang nicht vollständig ersetzen, sie dürfte jedoch ausreichend sein, um Verständnis für die Arbeit mit dBase III unter MS-DOS zu entwickeln und somit die effektive Ausnutzung und den Einsatz der 16-Bit-Technik wesentlich zu beschleunigen.
Die Nachnutzungsunterlagen umfassen etwa 100 Seiten. Teil I: Die Betriebssysteme CP/M und MS-DOS im Vergleich (mit Konfigurationshinweisen, Diskettenformaten, allen internen und externen MS-DOS-Befehlen und den Hilfsprogrammen), Teil II: Die Datenbanksysteme dBase II, dBase III und dBase III PLUS im Vergleich (mit Konfigurationshinweisen, einer Beschreibung der Prozeduren, Tips zum Datenaustausch mit Multiplan und WordStar sowie 13 Programmbeispielen), Teil III: Anhang (mit allen dBase III und MS-DOS-Befehlen der Versionen 2.11 und 3.1 in Beispielen).

VEB Optima Aschersleben, Wilhelmstraße 21/31, Aschersleben, 4320; Tel. 5151

Fischer

Erleichterter Dateizugriff beim MC 80

An der Sektion Informationstechnik der Technischen Universität Dresden wurde, aufbauend auf dem Grundsystem, ein Magnetbandbetriebssystem entwickelt, das durch Anlegen eines Datenverzeichnisses auf der Kassette den Zugriff auf eine Datei über die Angabe des jeweiligen Namens ermöglicht. Das Betriebssystem ist sowohl menügesteuert als auch durch Anwenderprogramme bedienbar. Es werden folgende Funktionen realisiert:
- Dateiverzeichnis von Kassette lesen/schreiben
- Dateiverzeichnis auf Bildschirm/Drucker
- Datei lesen/schreiben/löschen
- Kassette verdichten.

Vor jedem Schreib-/Lesezugriff erfolgt ein RAM-Kurztest. Das Betriebssystem benötigt etwa 3 KByte Programm- und 1 KByte Arbeitsspeicher.

Technische Universität Dresden, Sektion Informationstechnik, Bereich Kommunikations- und Computertechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4635121

Herzmann

Bilanzierung und Abrechnung in der Berufsausbildung

Das auf PC 1715 unter SCP/REDABAS laufende Programm umfaßt die Unterlegung aller mathematischen Zusammenhänge des Formblattes BA 001 mit Formeln. Nach Eingabe der mathematisch nicht erfaßten Größen erfolgt die komplette Bearbeitung und der Ausdruck des Formblattes.
Die Abarbeitung der Planung der Abt. Berufsbildung erfolgt innerhalb von 3 Stunden. Die Möglichkeit der Planänderung im laufenden Jahr ist (z. B. bei vorzeitigem Auslernen mehrerer Lehrlinge) gegeben.

VEB Funkwerk Kölleda, Eugen-Richter-Str., PSF 48, Kölleda, 5234

Erstellung technologischer Fertigungsunterlagen (TEFU)

Das Programm TEFU realisiert Aufgaben der technologischen Vorbereitung (technologische Fertigungsunterlagen).
Es besteht aus folgenden Komponenten:
- Erarbeitung von APSK/ABK
- Anzeige APSK/ABK (nach versch. Kriterien)
- Belegdruck (APSK/ABK, LS, MS)
- Hilfsprogramm zur Dateipflege und Systemverwaltung.

Bei Erstellung bzw. Änderung der APSK/ABK werden über Bildschirmmasken die entsprechenden Eingaben abgefordert (Kopf- und Arbeitsgangmaske). Der Anwender des Programmsystems wird durch eine umfangreiche Menüsteuerung und Kommentare durch diese geführt.
Hardware: PC 1715 (BC A5120/30) mit mind. 2 Diskettenlaufwerken/BAB II/beliebiger Drucker
Software: SCP-kompatibles Betriebssystem und REDABAS.

VEB Elektroprojekt und Anlagenbau Berlin, ZFT-RFT 2, Rhinstraße 100, Berlin, 1140

Wir suchen ...

... ein Sortierprogramm adäquat Super Sort bzw. mit ähnlichen Leistungsparametern zur Verarbeitung auf AC A 7100.

VEB Vereinigte Wellpappenbetriebe Leipzig, Werk 4, Rationalisierung Verpackung, Straße der Befreiung 17, Leipzig, 7010; Tel. 64156 (Koll. Borkenhagen)

Bindig

... nach einer Möglichkeit, einen Atari-Drucker als Peripheriegerät am KC 85/3 zu nutzen. Wer kann uns bei der Erstellung der Hard- und Softwarelösung behilflich sein?

Betriebsschule „W. M. Sagorski" Golzern, Bahren über Grimma, 7241, Tel. Grimma 670

Freitag

... dringend für Terminal CM 7202.M1 Beschreibung in Deutsch, auch leihweise.

VEB Komplette Chemieanlagen, ORZ, Abt. 695120, Postfach 184, Dresden, 8012; Tel. 4863323

Dr. Gäbler

Beiträge zur Mikrocomputertechnik

von M. Roth (Hrsg.), VEB Verlag Technik, Berlin 1986, 1. Auflage, 40,– M

Dieses Buch ist eine Sammlung von Aufsätzen und Vorträgen der 3. Jahrestagung „Mikroprozessortechnik '85" zu dem breiten Themenkreis Mikrorechentechnik. Dabei handelt es sich zum größten Teil um Übersichtsartikel, die von einzelnen Arbeiten, die tiefer in bestimmte Problemkreise eindringen, unterbrochen werden. Allen Artikeln gemeinsam ist die sehr hohe Informationsdichte. Zu vielen Schlagworten unserer Tage, wie 5. Computergeneration, 32-Bit-Mikroprozessor, Künstliche Intelligenz und Expertensystem, findet man vorausdenkende Artikel. Aktuelleren Bezug haben Arbeiten zu Rechnernetzen und zum Schaltkreisentwurf. Auf die Personalcomputer, die derzeit sicher die höchste Einsatzrate in der DDR haben, PC 1715 und A 7100, wird ebenso eingegangen wie allgemein auf Programmiersprachen und Betriebssysteme. Auch der aktuellen Problematik der Aus- und Weiterbildung wird gebührende Aufmerksamkeit gewidmet. Im Zusammenhang mit der Vernetzung von Computern wird ihre Kopplung untereinander und ihre Einbindung in ein industrielles Umfeld besprochen. Dabei zeigt sich, daß 8-Bit-Mikroprozessoren nach wie vor aktuell sind, sowohl für Personalcomputer mit begrenztem Aufgabengebiet als auch als Herzstück von Automatisierungsgeräten. Von fast philosophischer Tiefe ist der Artikel des Herausgebers, der sich mit dem urheberrechtlichen Schutz von Softwareprodukten befaßt. Abschließend sei noch bemerkt, daß das Buch trotz überblickartiger Artikel kein Einsteigerbuch ist, sondern bei vielen Beiträgen eine gewisse Vertrautheit mit der Materie voraussetzt.

M. Rachow

Praktische Mikrocomputertechnik

von M. Kramer, Militärverlag der DDR, Berlin 1987, 272 S., Reihe Amateurbibliothek

Die Mikrocomputertechnik hält in die verschiedensten Bereiche unseres Lebens Einzug. Gleichzeitig steigt das Interesse der Elektronikamateure an dieser Technik. Die *Praktische Mikrocomputertechnik*, erschienen in der Amateurbibliothek des Militärverlages, wendet sich vor allem an diesen Interessentenkreis. Der Autor bietet eine Anleitung zum Selbstbau und zur Programmierung eines eigenen Computers.

Im ersten Kapitel wird nach einer knappen Einleitung der konkrete Aufbau des Gerätes, das sind die Prozessorkarte, der Tastaturanschluß, der Anschluß eines Kassettenmagnetbandes und eines Fernschreibers (Druckers), die Bildschirmansteuerung, die Stromversorgung und der mechanische Aufbau behandelt. Im abschließenden Absatz *Inbetriebnahme* wird Anfängern geraten, sich von einem erfahrenen Amateur helfen zu lassen. Dieser Hinweis muß angesichts der gedrängten Darstellung unterstrichen werden.

Da der Amateur im wesentlichen ohne komfortable Meßtechnik arbeiten muß, sind die vorgeschlagenen Inbetriebnahmehilfsmittel von großer Bedeutung.

Die Programmierung des entstandenen Gerätes wird im zweiten Kapitel behandelt. Vorangestellt ist eine Einführung in die Programmiersprache BASIC. Anschließend werden die Maschinenbefehle des UB 880 D besprochen. Es folgen Erläuterungen der wichtigsten „Softwarewerkzeuge", wie Monitorprogramm und Debugger, Editor und Assembler sowie kurze Beispiele.

Spätestens an dieser Stelle muß auf den sehr umfangreichen Anhang verwiesen werden. Er macht mit 139 Seiten mehr als die Hälfte des Buches aus. In ihm sind die genannten Programme als komplette Assembler- und Interpreter und Disassembler, aber auch Hardwareunterlagen, wie Schaltkreisbeschreibungen und Leiterplattenlayouts machen den Anhang zu einer wichtigen Fundgrube für den Amateur. Nach der Programmierung werden im dritten Kapitel Speichererweiterungen, ein EPROM-Programmierzusatz, ein Zeitgeberanschluß, ein Magnetbandanschluß, ein Handleser für Lochstreifen sowie der Anschluß von D/A- bzw. A/D-Wandlern behandelt. Das vierte und letzte Kapitel enthält eine kurze Vorstellung einiger Peripheriebausteine, weiterer Mikroprozessoren und eine Abhandlung über den Anschluß eines Diskettenlaufwerkes, einschließlich einiger wichtiger Angaben über den Floppy-Disk-Schaltkreis U 8272, die, wie der Autor schreibt, noch kurz vor Redaktionsschluß entstand. Im Anhang findet sich dazu das Programm CBIOS. Wegen des gebotenen Umfanges sind dem Leser sicher manche Teile des Buches zu knapp geraten, und er kommt, will er der Aufforderung zum Selbstbau folgen, nicht ohne umfangreiche andere Literatur aus. Das Buch bietet jedoch neben vielen wertvollen Einzelinformationen und konkreten Hilfsmitteln einen Überblick und ist daher nicht nur für den Amateur interessant.

V. Obst

UNIX-Tabellenbuch

von F. Bach, C. Jansen und G. Spies, herausgegeben von F. Bach und P. Domann, Carl Hanser Verlag München Wien 1986

Eine im deutschen Sprachraum in Inhalt und Form wohl bisher einmalige Publikation zu UNIX-kompatiblen Betriebssystemkonzepten wurde vom Hanser Verlag mit dem UNIX-Tabellenbuch vorgelegt. Informationen, die bisher mühsam aus verschiedenen Quellen zusammengetragen werden mußten, findet der Leser hier übersichtlich geordnet und gut editiert. Berücksichtigung finden insbesondere die Standardversionen: UNIX Version 7, UNIX System III, UNIX System V Rel. 1, UNIX BSD 4.2 und XENIX 286 Rel. 3. Beginnend mit einem Systemüberblick der angegebenen UNIX-Standardversionen werden in dem UNIX-Tabellenbuch folgende Komplexe ausführlich behandelt:

1. *UNIX-Kommandosatz* mit einer alle Optionen umfassenden Kommandoerklärung.
2. *UNIX-Benutzeroberfläche* C-Shell und Bourne-Shell in einer tabellarischen Gegenüberstellung der verschiedenen Möglichkeiten.
3. *Dienstprogramm* 'awk' (Textmustererkennung und -verarbeitung), 'sccs' (Source Code Control System) und 'make' (Programmwartung und -generierung).
4. *Editieren* 'ed' (interaktiver Standard-Zeileneditor von UNIX) und 'vi' (interaktiver Bildschirmeditor/Cursor-orientiert).
5. *Textverarbeitung* mit einer Beschreibung und Kommandoübersicht des nroff/troff-Textverarbeitungssystems, die eine Behandlung von 'tbl' (Tabellenpräprozessor), 'eqn' (Formelaufbereitung) und der me-/ms-Makropakete einschließt.
6. *C-Programmierumgebung* mit der ausführlichen Behandlung der Komponenten 'ar' (Archivator), 'cb' (C-Formatierer), 'cc' (C-Compiler), 'lint' (C-Verifizierer) und 'nm' (Symboltabellenausgabe).
7. *UNIX-Systemcallbeschreibung* der verschiedenen Versionen.
8. *UNIX-Ein-/Ausgabekommandobehandlung* (cpio, dd, dump, dumpdir, restor, tar) und *UNIX-Superuser-Kommandos* einschließlich einer ausgezeichneten Beschreibung der *Termcap-Datenbasis* für die Ansteuerung unterschiedlicher Bildschirmterminals. Die Aufbereitung und Auswahl der Informationen des UNIX-Tabellenbuches erfolgte von den Autoren sehr sorgfältig und anwenderorientiert. Gesuchte Informationen lassen sich schnell und problemlos finden.

Dr. L. Claßen

Computertechnik im Profil

von H.-R. Schuchmann und H. Zemanek (Hrsg.) R. Oldenbourg Verlag München, 198 S.

Computergeschichte, lebendig in Gegenwart und Vergangenheit, gefüllt mit persönlichen Reminiszenzen bekannter und weniger bekannter Computertechniker und Informatiker – nach heutiger Terminologie – so stellt sich dieser Zeitbericht dar.

Die 35 Autoren von z. B. Bauer, Dijkstra, Kerner bis Zemanek und Zuse schreiben engagiert mit persönlicher Position fach- und sachkundig aber auch zugleich sozial verantwortungsbewußt und kritisch. Bei den ersten Relais- und Röhrenrechnern beginnend, überstreicht die Dokomentation alle allgemeinen Digitalrechner-Prinzipien, wie sie heute zu den ersten vier Generationen der J.-v.-Neumann-Architektur gerechnet werden können. Nur die Datenflußarchitektur geht partiell darüber hinaus. Alles, was „Geschichte" der Computertechnik in dieser Darstellung ist, ist „algorithmische Maschine" – vom „Ziffernrechner" bis zur „Datenverarbeitungsanlage", ist also alles, was man unstrittig als „nichtintelligente Rechenmaschine" nach der Algorithmus-Definition von A. M. Turing bezeichnen kann.

So ist es in der Tat ein Dokument der Vorgeschichte der Computertechnik, ihrer klassischen Epoche.

Prof. Dr. M. Roth

Fachtagung Computer- und Mikroprozessortechnik '87

Tagungsprogramm
Am 8. und 9. Dezember 1987 findet in Magdeburg die o. g. Tagung statt. Veranstalter sind die Wissenschaftliche Sektion Computer- und Mikroprozessortechnik des Fachverbandes Elektrotechnik der KDT sowie die Sektion Automatisierungstechnik und Elektrotechnik der Technischen Universität „Otto von Guericke" Magdeburg.
Die Fachtagung vermittelt umfassende Informationen über Stand und Perspektiven der Computer- und Mikroprozessortechnik, ihre Anwendung und aktuelle Probleme der Schaltkreisentwicklung unter besonderer Berücksichtigung der Anforderungen in der DDR.

Dienstag, den 8. Dezember 1987
(9.30 – 16.00 Uhr)

- *Eröffnung und Begrüßung*
- *Hauptvorträge*

P. Neumann; Magdeburg
Mikrorechner-Automatisierungsanlagen für die Mittel- und Großautomatisierung

M. Seifart, B. Michaelis; Magdeburg
Intelligente Funktionseinheiten zur Meßwerterfassung und Vorverarbeitung

M. Roth; Ilmenau
Mikroprozeßrechner und Expertensysteme – perspektivische Entwicklungen

P. Neubert; Dresden
Leistungsgesteigerte 16- und 32-Bit-Mikroprozessorsysteme

- *Sektion A: Meßwerterfassung und Prozeßautomatisierung*

G. Naumann; Dresden
Möglichkeiten der Kopplung von Meßtechnik mit Büro-/Personal- und Kleincomputern

Th. Schindler, U. Fincke, H. Beikirch; Magdeburg
Meßdatenverarbeitung mit feldbuskoppelbaren Rechnermodulen an Personalcomputern

H. Albrecht, A. Röhl, K.-P. Schulz; Rostock
Optoelektronisches Meßsystem in der Prozeßautomatisierung

R. Naumann, V. Schnetter, U. Frühauf; Dresden
Der Einsatz von Patterngeneratoren in einem Laborsystem zur Prüfung komplexer digitaler Leiterkarten

U. Manzelmann; Rostock
Computergrafik in der Prozeßautomatisierung

G. Richter, Dresden
Aspekte der Prüfstrategie und der prüfgerechten Erzeugnisgestaltung im Hinblick auf die Einführung der SMD-Technik

G. Schubert; Dresden
Erweiterungsbaugruppen zum Kleincomputersystem KC 87 für spezielle Anwendungen

- *Sektion B: Mikrorechner (Hard- und Software)*

B.-G. Münzer, N. Wengel; Rostock
16-Bit-Single-Board-Computer WPU 8086 und 8000

L. Claßen; Berlin
P 8000 – Ein Programmier- und Entwicklungssystem mit UNIX-kompatibler Programmierumgebung

T. Schwenke; Berlin
Einsatz von 16-Bit-Rechnern im Verkehrswesen der DDR

K. Bernstein; Leipzig
Softwarezuverlässigkeit und ihre Behandlung

H. Schoenyan; Dresden
Hardware-Konzept des ESER-PC EC 1834

D. Wiedemuth; Dresden
Konzept der Basis-Software des EC 1834

N. A. Shutova u. a.; Dubna
Programmsystem zur Meßwerterfassung und Übertragung in der Experimentautomatisierung

Mittwoch, den 9. Dezember 1987
(8.00 – 13.00 Uhr)

- *Hauptvorträge*

W. Meiling; Dresden
Der Einfluß der Prozessorarchitektur auf die Struktur von Rechnerbussystemen

H. Löffler; Dresden
Lokale Netze

D. Müller; Karl-Marx-Stadt
Entwurf und Anwendung von Gate-Array-Schaltkreisen unter dem Aspekt der Zusammenarbeit von Anwender und Hersteller

M. Gieseler; Dresden
Entwurf und Anwendung von Gate-Array-Schaltkreisen unter dem Aspekt der mit U 5200 erreichten technischen Parameter

- *Sektion C: Bussysteme und ihre Anwendung*

F. Güttler; Magdeburg
Feldbussystem für die Kleinautomatisierung

W. Fengler, U. Steubel, N. Hirt; Ilmenau
Mehrschichtenmodell des Prozeß-LAN ISB 881

W. Kriesel, P. Gibas, Ch. Kozub; Leipzig
Funktionelle Maßnahmen zur Realisierung eines fehlertoleranten Feldbussystems

O. I. Elisarov, S. Günther, B. E. Resaev, K.-H. Schultz; Dubna
Intelligentes Terminal für das Rechnernetz des LNF am VIK-Dubna

G. Wollenberg, C. Pyttlik; Magdeburg
Eine zentrale Busvergabesteuerung für 16-Bit-Multimikrorechnersysteme

J. Hähniche, D. Kuhn; Magdeburg
Pilotprojekt des LAN an der TU „Otto von Guericke" Magdeburg

- *Sektion D: Kundenspezifische Schaltkreise*

U. Donath, G. Kurth, P. Schwarz, T. Trappe; Dresden
KOSIM-LT: Ein Logik- und Timing-Simulator für 16-Bit-Rechner

D. H. Eckhardt; Dresden
Entwicklungsrichtungen bei Siliconcompilern

K. Benning; Dresden
Anwendungsbeispiele und Anwendererfahrungen mit dem Gate-Array-System U 5200

B. Hertwig; Magdeburg
Einsatzbeispiel für das Gate-Array-System U 5200 im Schwermaschinenbau

H.-J. Ahrend, B. Stitz; Magdeburg
Ein Gate-Array-Schaltkreis für die Überwachung und Steuerung von Gasbrennern

B. Lux, D. Köhler, H. Helms, M. Steimann; Dresden
Berührungslose Fertigungsmeßtechnik mit CCD-Zeilen

- *Posterbeiträge*

H. Petzold, J. Krone, P. Holfert, A. Staudinger; Dresden
Qualitätssicherung mit Mehrstellenmeßtechnik

F. Eichelbaum; Magdeburg
Programmierbarer Meßverstärker mit verkürzter Einschwingzeit

W. Fengler, N. Hirt, D.-Q. Mink, G. Schade; Ilmenau
PI-Schrittreglermodul mit Einchip-Mikrorechner

J. Grünwald; Rostock
Mikrorechnergestützte Steuerung von Kleinantrieben

K. G. Rodinov, B. N. Solovjev, V. G. Tishin; Dubna
Mikroprozessor-Interface zur Steuerung eines Motors für einen Neutronenstrahl-Chopper

S. Günther, O. I. Elisarov, M. Löbner, B. Michaelis, A. I. Ostrovnoj, W. Schwenkner, K.-H. Schultz; Dubna
Dezentrales System zur Automatisierung des Spektrometers SPN-1 im VIK-Dubna

B.-G. Münzer, G. Jarke; Rostock
Multirechnersysteme mit 16- und 8-Bit-Rechnern

P. Eichelbaum; Magdeburg
Kopplung verschiedener Mikrorechnertypen mit Zwei-Tor-Speichern

A. Gienapp, B.-G. Münzer, W. Wengel; Rostock
Computergestützte Ausbildung in der Elektronik

H. Zimmerling; Leipzig
Datenbanksysteme und ihre Hierarchie auf Personalcomputern

V. V. Trofimov, u. a.; Dubna
Programm zur Realisierung des Datenaustauschs zwischen ESER-Rechnern und Mikrorechnern

O. Krohmann; Magdeburg
Die Nutzung von verbindungslosen Link-Diensten zur Implementation von Anwendersoftware für LANs am Beispiel eines Filetransferprotokolls

A. Eckstädt; Magdeburg
Simulation von Bussystemen und Konsequenzen für die Gestaltung künftiger Systeme

G. Roche, W. Ulbrich, M. Roth; Ilmenau
Worst-Case-Systembus-Prüftechnik

P. Schwarz; Dresden
Modellierung und Simulation von Gate-Array-Schaltkreisen

R. Maaß, J. Krieger, H. Hocker; Magdeburg
Gate-Array-Schaltkreis zur Signalauswertung in Meßsystemen mit CCD-Zeilen

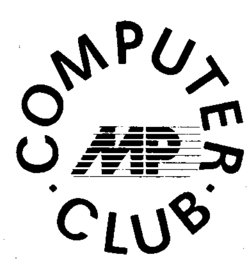
Das Tagungsbüro befindet sich am 7. Dezember 1987 von 14 bis 20 Uhr im Hotel International und während der Tagung am Tagungsort. Tagungsorte sind die Scala-Lichtspiele Magdeburg, Halberstädter Straße 135, und das Theater der Freundschaft Magdeburg, Braunschweiger Straße 25.

✉ KONTAKT ☎

Anfragen sind zu richten an:

(organisatorisch)
Kammer der Technik, Tagungsbüro
Brüderstraße 3
Stendal
3500
Tel. Stendal 21 30 34

bzw. (inhaltlich)
Technische Universität
„Otto von Guericke" Magdeburg
Sektion Automatisierungstechnik
und Elektrotechnik
Wissenschaftsbereich Prozeßmeßtechnik
Prof. Dr. sc. techn. Seifart
oder Doz. Dr. sc. techn. Michaelis
Boleslav-Bierut-Platz 5, PSF 124
Magdeburg
3010
Tel. Magdeburg 59 20



Liebe Leser,

nach dem Start unserer neuen Rubrik MP-Computerclub im Septemberheft haben uns bereits zahlreiche Zuschriften erreicht, für die wir uns auch im Namen der Leser bedanken, die aus den Beiträgen den gewünschten Nutzen ziehen werden.
Wir möchten uns an dieser Stelle noch einmal an alle potentiellen Autoren wenden, um auf die Beschaffenheit der Manuskripte hinzuweisen, deren Form eine zügige und qualitätsgerechte Veröffentlichung beeinflußt. Ausführliche Hinweise zum Abfassen von Manuskripten hatten wir bereits in MP 2/1987 abgedruckt, Probleme zu dieser Thematik in MP 9/1987.
Für das Veröffentlichen von Manuskripten im MP-Computerclub ist vor allem folgendes wichtig:

- *Zum Abdruck der Programme wird die Vorlage des Autors als Original genutzt. Daher liefern Sie bitte unbedingt weißes Papier mit kontrastreicher Schrift.*
- *In der Regel sollten die Programme als Quelltext-Listing, nur nach Absprache mit der Redaktion als Hexdump, eingesandt werden.*

Ihre Redaktion MP

KC85/3-BASIC-Tip Funktionsdefinition im Programm

Der BASIC-Interpreter gestattet es dem Programmierer, selbst Funktionen mit der DEF-FN-Anweisung zu definieren. Dies ist aber nur bei der Programmeingabe bzw. im EDIT-Modus möglich, das heißt, nicht im laufenden Programm. Ist dies gewünscht, z. B. in Programmen, die auch von anderen Nutzern bedient werden sollen, kann das Programm in Bild 2 helfen.
Es wird durch ein Maschinenprogramm unterstützt, welches eine im Programmlauf eingegebene Funktion in die interne Darstellung des Interpreters umwandelt und ins Programm „pokt".
Um die Nutzung einer Funktionseingabe zu demonstrieren, wird im Programm eine mathematische Funktion grafisch dargestellt. Die Koordinatenachsen werden ebenfalls eingezeichnet. Es erfolgt allerdings keine Anpassung bezüglich des dagestellten Maßstabs. Dies auszubauen, bleibt dem Anwender überlassen. Das Programm ist als Anregung zu eigenen Programmen gedacht und erhebt nicht den Anspruch auf optimale Realisierung.
Wichtig ist hier nur die Darstellung der Funktionseingabe. Man geht dabei folgendermaßen vor:

```
10 !.................................................
....................
20 DEFFNY(X)=X:!-----**************************************
*******************
30 WINDOW0,31,0,39:CLS
40 LINE0,128,319,128:LINE160,0,160,255
50 LOCATE1,1
60 A$="":INPUT" F(X) = ";A$
70 IFA$=""THEN60
80 CALL*406
90 FORX=-10TO10STEP0.05:Y=FNY(X)
100 PSET160+15*X,128+8*Y,7
110 NEXT
120 GOTO50
```

2

```
%DISPLAY 400 480 4
0400 00 4D 04 0A 00 90 FE E5 _M______
0408 D5 C5 21 62 03 E5 CD DA __!b____
0410 C4 E1 11 58 04 AF BE 28 ___X____
0418 04 ED A0 18 FB EB 36 3A ______6:
0420 23 36 8E C1 D1 E1 F1 C9 #6______
0428 2E 2E 2E 2E 2E 2E 2E 2E ........
0430 2E 2E 2E 2E 2E 2E 2E 2E ........
0438 2E 2E 2E 2E 2E 2E 2E 2E ........
0440 2E 2E 2E 2E 2E 2E 2E 2E ........
0448 2E 2E 2E 2E 00 97 04 14 ....____
0450 00 94 A7 59 28 58 29 B4 ___Y(X)_
%
```

1

1. Eingabe des Programmes gemäß Bild 2
Die Zeilen 10 und 20 werden in maximaler Länge mit Füllzeichen eingegeben.
Die Zeile 10 soll das Maschinenprogramm aufnehmen (vgl. /1/). Zeile 20 nimmt die jeweils eingegebene Funktion auf, welche unterschiedlich lang sein kann. Deshalb wird die Funktion auch stets mit einem Doppelpunkt abgeschlossen, und der Rest der Zeile wird dann mit einem Ausrufezeichen als Kommentar abgetrennt.
2. Verlassen des BASIC-Interpreters mit BYE
3. Anzeige des Speicherbereichs des BASIC-Programmes mit DISPLAY (Bild 1) und Übergang in die MODIFY-Eingabe mit der STOP-Taste
Die Eingabe erfolgt hexadezimal ab Adresse 406H bis 427H. Die Zeilen müssen dabei einzeln mit der ENTER-Taste abgeschlossen werden. Nach Verlassen des Eingabemodus wird der Warmstart des BASIC-Interpreters aufgerufen, und das Programm kann getestet werden.

K.-D. Kirves

Literatur

/1/ Kirves, K. D.: KC85/3-BASIC-Tip Maschinenprogramme (1). Mikroprozessortechnik, Berlin 1 (1987) 11

Primzahl-Nachlese zum BASIC-Rundfunklehrgang

Nachdem die ersten Programme der Rundfunkhörer zu den Primzahlzwillingen ausgewertet waren, erreichte mich noch eine Vielzahl von interessanten Lösungen. Davon möchte ich hier – ergänzend zu Heft 8/87, S. 283 drei vorstellen.
Durch die vielen DATA-Werte und eine komplexere Programmierung erreicht M. Reisener (Bild 1) etwa die vierfache Geschwindigkeit wie bei meinem Bezugsprogramm. Je die etwa dreifache Geschwindigkeit haben die anderen beiden Programme, wobei allerdings ihr Zeitverhalten unüblich ist. So speichert das Programm von K.-P. Asmus (Bild 2) jeweils erst einmal etwa 5000 Werte, um diese dann geschlossen auszugeben. Beim Programm von J. Schur (Bild 3) – was ich besonders interessant finde, nicht nur wegen seiner Kürze – wird ein Feld aufgebaut. Es beginnt, dadurch bedingt, ganz langsam und verzögert mit den ersten Werten; dann wird es immer schneller.
Es ist m. E. wirklich lohnenswert, diese und die anderen Programme zu analysieren. Man lernt dabei recht viel.
Vielen Dank allen Einsendern.

Prof. Dr. H. Völz

```
0 REM Manfred Reisener, Salzwedel
10 DIM H(15),S(48),P(250)
20 DATA 12,6,12,12,18,12,30,6,30,12,18,12,12,6,12
30 DATA 10,2,4,2,4,6,2,6,4,2,4,6,6,2,6,4,2,6,4,6,8,4,2,4
40 DATA 2,4,8,6,4,6,2,4,6,2,6,6,4,2,4,6,2,6,4,2,4,2,10,2
50 RESTORE 20: FOR I=1 TO 15: READ H(I): NEXT
60 RESTORE 30: FOR I=1 TO 24: READ S(I): NEXT
70 RESTORE 40: FOR I=25 TO 48: READ S(I): NEXT
80 N=-1: Q=49: L=0: M=-1: Y=1: P(1)=7: R=49
90 PRINT 3,5: PRINT 5,7
100 FOR Z=1 TO 1E6 STEP 0: FOR I=1 TO 15
110 Z=Z+H(I): IF Z<Q THEN 170
120 L=L+1: Y=Y+S(L): IF L=48 THEN L=0
130 IF Y>=R THEN M=M+1: R=P(M+1)*P(M+1)
140 FOR J=1 TO M: W=Y/P(J)
150 IF W=INT(W) THEN J/M: NEXT: GOTO 120
160 NEXT: N=N+1: P(N+1)=Y: Q=Y*Y
170 FOR K=1 TO N: W=(Z-2)/P(K)
180 IF W=INT(W) THEN K=N: NEXT: NEXT: NEXT: END
190 NEXT: FOR K=1 TO N: W=Z/P(K)
200 IF W=INT(W) THEN K=N: NEXT: NEXT: NEXT: END
210 NEXT: PRINT Z-2,Z: NEXT: NEXT: END
READY
```

1

```
10 REM Juergen Schur, Dresden
20 DIM A(5501): X=1
30 FOR I=3 TO 5500 STEP 2
40 IF A(I)=1 THEN NEXT
50 IF X+2=I THEN PRINT X,I
60 X=I
70 FOR J=I TO 5500 STEP 2*I
80 A(J)=1: NEXT
90 NEXT
READY
```

3

```
0 REM Klaus-Peter Asmus, Auerbach
10 W=0: X=0: S=0: M=500: N=3125
20 DIM Z(N), P(M): Z(1)=1: FOR W=1 TO M: P(W)=3*W+1: NEXT
30 FOR R=1 TO 160: T=Y: Y=R*N
40 FOR W=1 TO N: Z(W)=(T+W)*2+1: NEXT: S=INT(SQR(Z(N))/2-1)
50 FOR X=1 TO S: FOR W=P(X) TO N STEP 2*X+1:Z(W)=1: NEXT
60 P(X)=W-N: NEXT
70 FOR W=S+1 TO M: X=2*W+1: P(W)= P(W)+X*INT((N-P(P))/X+1)-N: NEXT
80 B=Z(N): FOR W=1 TO N
90 IF Z(W)=1 THEN W=W+1: NEXT: GOTO 110
100 IF Z(W-1)>1 THEN PRINT Z(W-1),Z(W): ELSE NEXT
110 Z(0)=B: NEXT
READY
```

2

Bildausschnittverdopplung

Das vorgestellte Maschinenprogramm verdoppelt beim KC 85/(/3) die Zeilen 0 bis 15 und die Spalten 0 bis 15. Das Programm wird auf den Adressen 0000 bis 0103 abgelegt. Da dieser Speicherbereich nur bedingt frei ist, kann es passieren, daß Adressen überschrieben werden. Danach ist das Programm zu korrigieren oder neu einzuladen. Um das Programm nicht unnötig zu komplizieren, wurde auf die Spalten 16 bis 19 verzichtet. Das Programm wird aus dem BASIC-Modus über CALL 0 aufgerufen. Es ist von Vorteil, das maximale Fenster einzustellen. Achtung! Das Programm belegt während der Abarbeitung jeweils 16 Adressen beginnend bei BA00, BB00, BC00, BD00 und BE00.
Für einen Test kann das Programm in Bild 2 genutzt werden.

E. Feige

```
BILDAUSSCHNITTVERDOPPLUNG

ADR.
0000 CD 1B F0 21 D4 00 11 00 BA 01 10 00 ED B0 11 00
0010 BB 0E 10 ED B0 11 00 BD 0E 10 ED B0 DD E5 FD E5
0020 16 8F 1E 9F 06 08 D5 C5 21 00 BC 06 04 72 2C 72
0030 2C 15 72 2C 72 2C 14 BD 10 F3 21 00 BE 06 04 73
0040 2C 1D 73 2C 1C 10 F8 1D 1D 06 04 73 2C 1D 73 2C
0050 1C 10 F8 06 10 0E 00 26 BE 69 56 25 5E DD 21 00
0060 00 FD 21 80 00 DD 19 FD 19 25 56 25 5E EB 0C C5
0070 06 10 7E 08 7E CB BF CB B7 CB AF CB A7 0E 00 B9
0080 CA 87 00 0C C3 7F 00 E5 26 BA 69 56 08 CB 3F CB
0090 3F CB 3F CB 3F 0E 00 B9 CA 9F 00 0C C3 97 00 69
00A0 5E DD 73 00 DD 72 01 FD 73 00 FD 72 01 E1 23 DD
00B0 23 DD 23 FD 23 FD 23 10 B9 C1 10 9B C1 D1 15 15
00C0 1D 1D 1D 1D C3 CA 00 C3 26 00 10 FB FD E1 DD E1
00D0 CD 1B F0 C9 00 03 0C 0F 30 33 3C 3F C0 C3 CC CF
00E0 F0 F3 FC FF E0 60 E0 60 C0 40 C0 40 A0 20 A0 20
00F0 80 00 80 00 60 60 40 40 20 20 00 00 60 60 40 40
0100 20 20 00 00
```

Bild 1 Programm Bildausschnittverdopplung

```
TEST-PROGRAMM

10 WINDOW 0,31,0,39:CLS
20 FOR X=0 TO 25
30 ?AT(0,0); CHR$(65+X)
40 FOR Y=0 TO 4
50 CALL 0:PAUSE 5
60 NEXT Y:NEXT X:END
```

Bild 2 Testprogramm

KC83/3-Assemblertip Zusatzprogramm zur Steuerung der Listenausgabe

Der KC85-Modul MO27-DEVELOPMENT (vgl. /1/) besitzt die Möglichkeit der Ausgabe des Assemblerlistings auf Drucker. In manchen Fällen ist es aber nicht erforderlich, stets das komplette Listing auszudrucken. Der Assembler beinhaltet jedoch keine Pseudoanweisung zur Listensteuerung. Das Programm in Bild 1 kann diese Funktion ergänzen.
Das EDAS-Untermenü arbeitet prinzipiell genauso wie das CAOS-Grundmenü, es ist in /2/ beschrieben. Im Gegensatz zum CAOS-Menü werden als Prolog zwei Bytes DDH verwandt. Der Suchbereich für EDAS-Menüanweisungen erstreckt sich auch auf den Bereich von BAOOH bis BFFFH. Das in Bild 1 dargestellte Programm enthält zwei Menüanweisungen zum Ein- und Ausschalten der Listensteuerung. Weiterhin ist ein Seitenvorschub im Quellprogramm programmierbar – zum Erreichen eines übersichtlicheren Listings. Die Listensteuerung erfolgt durch Eintragen von drei speziellen Steuerzeichen in Kommentarzeilen des Quellprogrammes. Die Eingabe der Steuerzeichen erfolgt über die Funktionstasten F4 bis F6, ergänzend zu den vom EDAS verwendeten Funktionstasten. Tafel 1 zeigt die Funktionstasten, Codes, Darstellung und ihre Bedeutung.
Bei Eingabe des Kommandos LISTOFF wird die Listensteuerung aktiviert, indem ein Programmteil zur Auswertung der Steuerzeichen zwischen Assembler und Druckertreiber „geschaltet" wird. Nach Ausdruck des Assemblerlistings sollte mit der Anweisung LISTON die normale Druckerausgabe wieder eingeschaltet werden, da ansonsten eine unterdrückte Ausgabe auch für den Disassembler oder die DUMP-Ausgabe bestehen bleibt. Ebenfalls sollte stets am Ende des Quellprogrammes das Steuerzeichen zum Einschalten der Ausgabe (F4) stehen.
Das in Bild 1 dargestellte Programm ist mit dem Modul MO27 DEVELOPMENT erstellt worden. Es ist sowohl auf dem KC 85/3 als auch auf dem KC 85/2 lauffähig. Es arbeitet, außer mit den im Modul MO27 enthaltenen Druckertreibern, mit allen Druckertreibern zusammen, die über den USER-Kanal 2 arbeiten. Das Programm kann auch auf andere Speicherbereiche übersetzt werden, es ist jedoch nur im Bereich BAOOH bis BFFFFH nutzbar.

K.-D. Kirves

Tafel 1 Aufstellung der Steuerzeichen

F-Taste	Code	Zeichen	Bedeutung
F3	9CH	umrandetes L	Unterdrückung der Listenausgabe
F4	8FH	HARDCOPY-Zeichen	Freigabe der Listenausgabe
F5	8CH	CLS-Zeichen	Seitenvorschub

```
0000            ;EDAS-ZUSATZPROGRAMM ZUR LISTENSTEUERUNG
0000            ;Created by K.-D. KIRVES
0000            ;          VEB MOM
0000            UOUT1   EQU     0B7EDH  ;LISTKANAL
0000                    ORG     0BF00H
BF00            ;LISTENSTEUERUNG EINTRAGEN
BF00 0000               DEFW    0000DH
BF02 4C495354           DEFM    'LISTOFF'
BF09 01                 DEFB    1
BF0A 3A6ABF             LD      A,(MERK)        ;LIST?
BF0D A7                 AND     A               ;=0?
BF0E C8                 RET     Z               ;->ENDE
BF0F AF                 XOR     A
BF10 326ABF             LD      (MERK),A        ;(MERK)=0
BF13 2ABEB7             LD      HL,(UOUT1+1)    ;AKT.AUSG.
BF16 2268BF             LD      (MOUT),HL       ;MERKEN
BF19 2256BF             LD      (PR1),HL        ;IN PROGR.
BF1C 214BBF             LD      HL,NOUT         ;NEUE AUSG.
BF1F 22BEB7             LD      (UOUT1+1),HL    ;EINTRAGEN
BF22 110599             LD      DE,0B905H       ;F-TASTEN
BF25 2162BF             LD      HL,FTAB         ;NEUE BEL.
BF28 010600             LD      BC,6
BF2B EDB0               LDIR
BF2D C9                 RET
BF2E            ;LISTENSTEUERUNG RUECKSETZEN
BF2E 0000               DEFW    0000DH
BF30 4C495354           DEFM    'LISTON'
BF36 01                 DEFB    1
BF37 3A6ABF             LD      A,(MERK)        ;LIST?
BF3A A7                 AND     A               ;<>0?
BF3B C0                 RET     NZ              ;->ENDE
BF3C 3E01               LD      A,1
BF3E 326ABF             LD      (MERK),A        ;(MERK)=1
BF41 2A68BF             LD      HL,(MOUT)       ;GEMERKTES
BF44 7D                 LD      A,L
BF45 B4                 OR      H               ;WERT VORH?
BF46 C8                 RET     Z               ;NEIN ->
BF47 22BEB7             LD      (UOUT1+1),HL    ;RUECKSCHR.
BF4A C9                 RET
BF4B            ;NEUES AUSGABEPROGRAMM
BF4B FE9C       NOUT:   CP      9CH             ;LISTOFF?
BF4D 2809               JR      Z,N1
BF4F FE8F               CP      8FH             ;LISTON?
BF51 280B               JR      Z,N2
BF53 E67F               AND     7FH
BF55 C30000     PR2:    JP      0               ;->AUSG.ALT
BF58            PR1     EQU     $-2             ;=ADRESSE
BF58 3EC9       N1:     LD      A,0C9H          ;= RET
BF5A 3255BF     N3:     LD      (PR2),A         ;EINTRAGEN
BF5D C9                 RET
BF5E            ;
BF5E 3EC3       N2:     LD      A,0C3H          ;= JP
BF60 18F8               JR      N3
BF62            ;FUNKTIONSTASTENBELEGUNG
BF62 9C00       FTAB:   DEFW    9CH             ;F3=LISTOFF
BF64 8F00               DEFW    8FH             ;F4=LISTON
BF66 8C00               DEFW    8CH             ;F5=EJECT
BF68            ;
BF68 0000       MOUT:   DEFW    0               ;MERK-ADR.
BF6A 01         MERK:   DEFB    1               ;MERK-LIST
ERRORS: 0000
```

Bild 1 Assemblerlisting zum Programm Listensteuerung

Literatur

/1/ Domschke, W.: Das Softwarekonzept des KC 85/3. Mikroprozessortechnik, Berlin 1 (1987) 3, S. 89–91

/2/ Kirves, K.-D.: Modul M027 Development – Assemblerprogrammierung für KC 85/3. Mikroprozessortechnik, Berlin 1 (1987) 8, S. 247–249

ISSN 0232-2892
Mikroprozessortechnik, Berlin 1 (1987) 11
S. 321–352

Arbeitsplatz für Konstruktion und Technologie A 6454

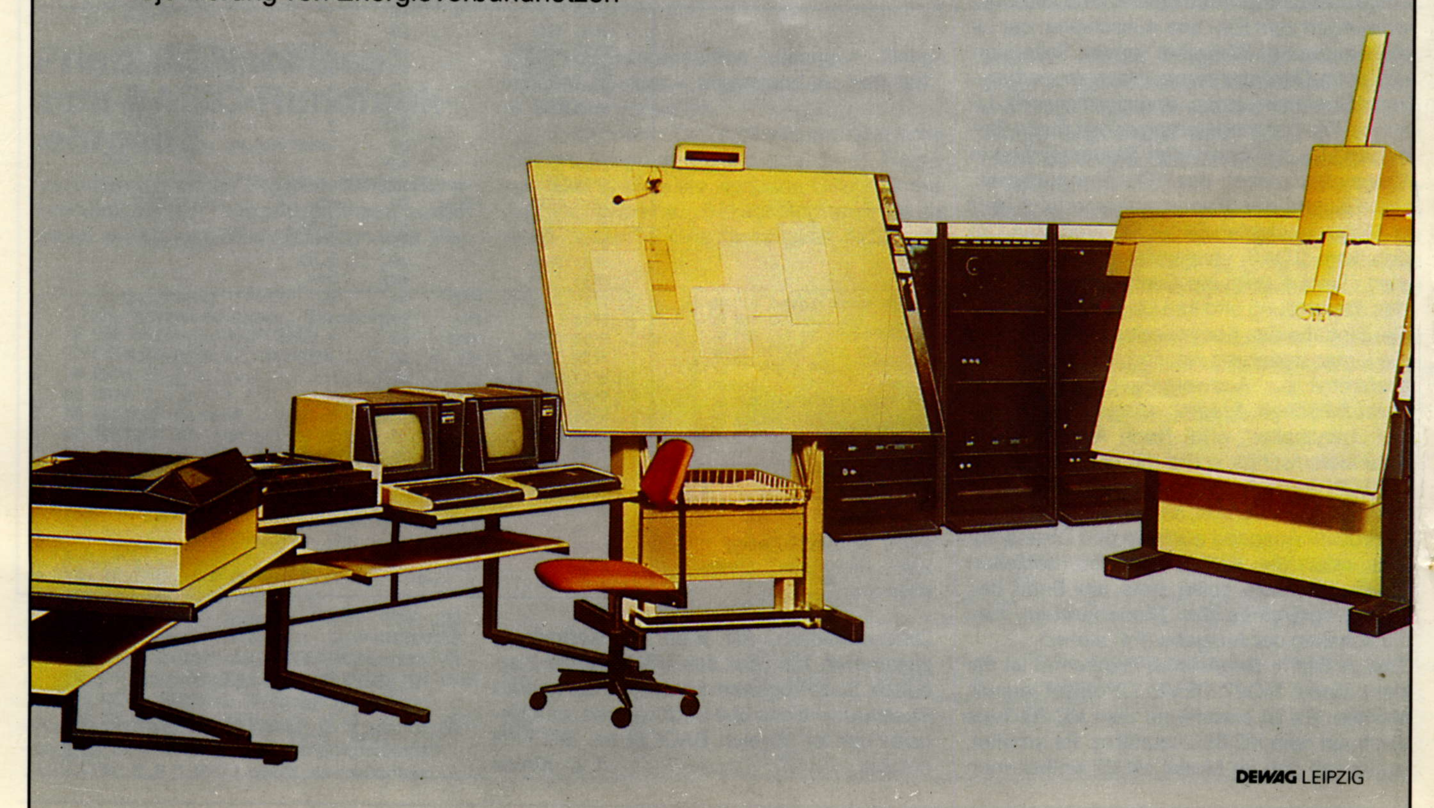
Der Arbeitsplatz robotron A 6454 ist ein interaktives grafisches System zur Rationalisierung der Entwurfs-, Konstruktions-, Technologie-, Fertigungs- und Projektierungsaufgaben mit einem oder mehreren Arbeitsplätzen sowie Kopplung an über- oder nebengeordnete Systeme. Das modular aufgebaute CAD/CAM-System bietet ein breites Spektrum von Anwendungsmöglichkeiten:

- Rechnergestütztes Herstellen von Zeichnungen
- Rechnergestütztes Programmieren von Maschinen und Anlagen
- Rechnergestütztes Ausarbeiten von APSK, Stücklisten, Arbeitsunterweisungen für Maschinenbedienung
- Berechnen von Einzelteilen
- Rechnergestützte Arbeitsplätze für Konstrukteure und Technologen
- Rationalisierung der Projektierung und Offertentätigkeit im Großmaschinen- und Anlagenbau
- Entwicklung von Gleichstrom-Stellmotoren
- Entwicklung und Herstellung von Leiterplatten
- Projektierung von Energieverbundnetzen

robotron

VEB Robotron-Vertrieb Berlin
Mohrenstraße 62
Berlin
DDR – 1080

Exporteur:
Robotron Export-Import
Volkseigener Außenhandelsbetrieb
der Deutschen Demokratischen Republik
Allee der Kosmonauten 24
Berlin
DDR – 1140



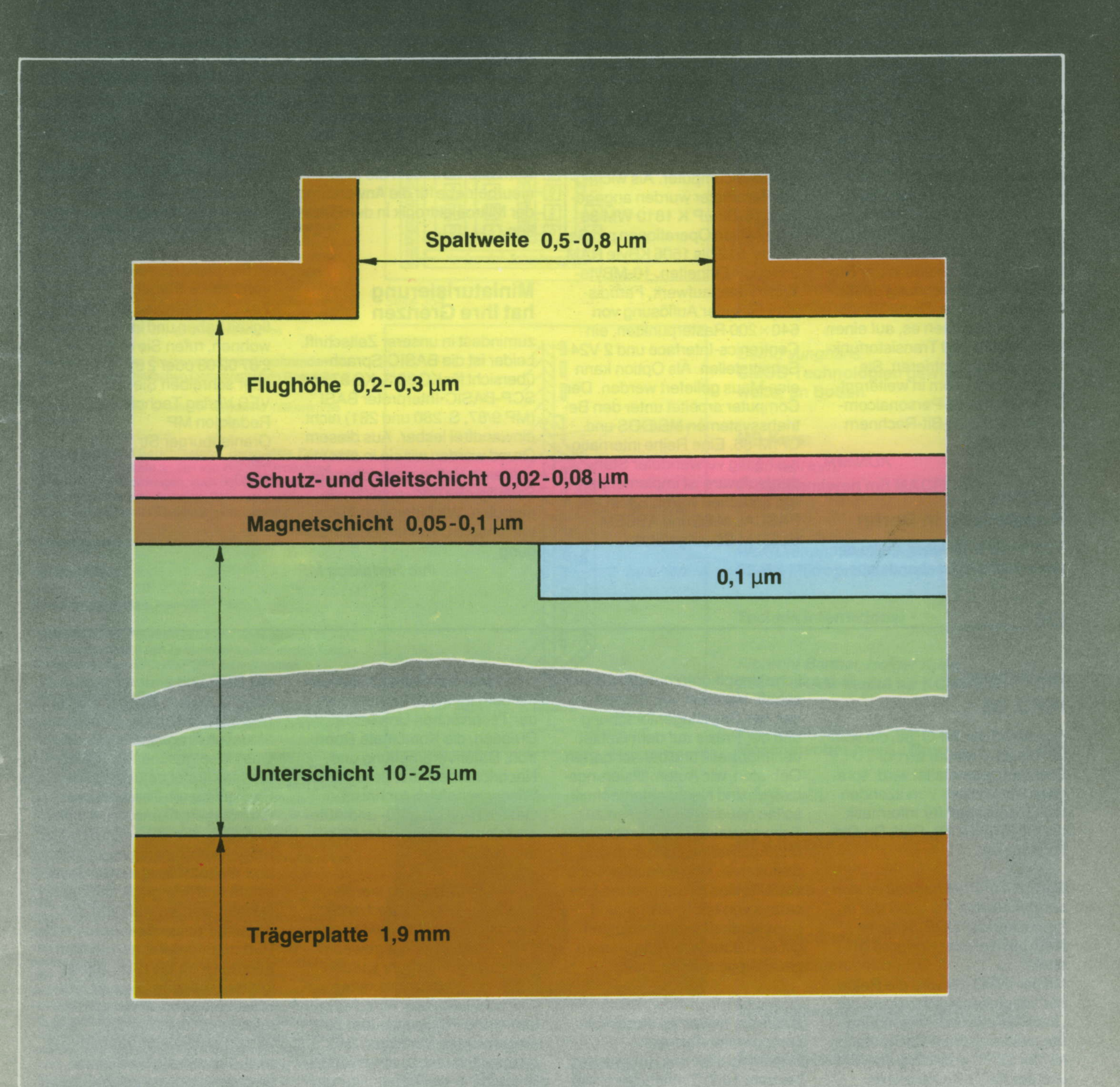
ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 12 · 1987

Mikroprozessortechnik

VEB Verlag Technik Berlin
ISSN 0232-2892



Speichertechniken

- 32-Bit-Prozessoren
- Einchipmikrorechner

256-Kilobit-Speicherschaltkreise aus eigener Produktion

In der Volkswirtschaft der DDR wurde am Vorabend des 38. Jahrestages dazu übergegangen, 256-Kilobit-Speicherschaltkreise aus eigener Produktion anzuwenden, nachdem eine entsprechende technologische Linie im Kombinat Carl Zeiss JENA ihren Betrieb aufgenommen hat. Eine moderne Technologie (CMOS-Technik) erlaubt, Strukturbreiten von 1,5 μm zu beherrschen. Auf deren Grundlage ist es möglich, weitere Schaltkreistypen zu produzieren und sich auf die Produktion von 1- bzw. 4-Megabit-Speicherschaltkreisen mit konkreten Schritten vorzubereiten. 256-Kilobit-Speicherschaltkreise stellen ein qualitativ höheres Niveau in der Produktion und Anwendung der Mikroelektronik dar. Diese Schaltkreise ermöglichen es, auf einen Chip rund 60 000 Transistorfunktionen zu konzentrieren. Sie kommen vor allem in weiterentwickelten 16-Bit-Personalcomputern sowie 32-Bit-Rechnern zum Einsatz.

ADN/MP

ELORG – Ausstellung in Berlin

Vom 6. bis 9. Oktober stellte der sowjetische Außenhandelsbetrieb ELORG in der ständigen Exportmusterschau im Haus der sowjetischen Wissenschaft und Kultur ausgewählte Exponate seines Handelssortiments vor. So wurden Bauelemente, Taschenrechner, Schulcomputer und der Drucker EC 7040 ausgestellt.
Auf Schautafeln wurde über weitere Erzeugnisse informiert. Beispielsweise über den Großrechner EC 1066, das leistungsfähigste Modell des ESER, über den Terminalrechner EC 1007 und über den Personal- und Arbeitsplatzcomputer EC 1841. Beim EC 1841 handelt es sich um einen IBM-PC/XT-kompatiblen Personalcomputer. Als wichtigste Parameter wurden angegeben: 16-Bit-μP K 1810 WM 86 mit 1 Million Operationen pro Sekunde, 512 bis 1506 KByte RAM, 2 Floppy-Einheiten, 10-MByte-Winchesterlaufwerk, Farbdisplay mit einer Auflösung von 640×200 Rasterpunkten, ein Centronics-Interface und 2 V24-Schnittstellen. Als Option kann eine Maus geliefert werden. Der Computer arbeitet unter den Betriebssystemen MS/DOS und CP/M-86. Eine Reihe international häufig verwendeter Standardsoftware ist implementiert. Als Sprachen waren BASIC-86, PASCAL M 86 und ASSEMBLER M-86 aufgeführt.

MP

Berichtigung

Leider hatte in MP 9/87 auf Seite 250, 3. Spalte oben, der Druckfehlerteufel zugeschlagen. Richtig muß es heißen:
... sei auf die Bezugsmöglichkeiten von Angebotsrecherchen aus dem ...
und AR 30/87 ME-Einsatz zur rationellen Energieanwendung.
Außerdem bittet der VEB Applikationszentrum Elektronik Berlin noch um folgenden Zusatz: Direktbestellungen der Angebotsrecherchen sind zu richten an den VEB Applikationszentrum Elektronik Berlin, Abt. DA, Mainzer Str. 25, Berlin, 1035. Weitere Kontakte vermitteln die Ingenieurbetriebe für die Anwendung der Mikroelektronik in den Bezirken.

MP

Miniaturisierung hat ihre Grenzen

zumindest in unserer Zeitschrift. Leider ist die BASIC-Sprachübersicht für KC 85/3, KC 87 und SCP-BASIC-Interpreter BASI (MP 9/87, S. 280 und 281) nicht einwandfrei lesbar. Aus diesem Grund werden wir sie in einem späteren Heft in ausreichender Schriftgröße und -qualität wiederholen. Wir bitten den Autor und unsere Leser um Entschuldigung.

Ihre Redaktion MP

In eigener Sache

Ab sofort suchen wir für die Stelle eines Redakteurs unserer Zeitschrift MP eine(n) geeignete(n) Mitarbeiter(in) mit abgeschlossenem Hoch- oder Fachschulstudium und guten Kenntnissen der Computertechnik (Hard- und Software).
Zu den Aufgaben gehören:
– Betreuen des Sachgebietes Computertechnik in der Zeitschrift
– Gewinnen und redaktionelles Bearbeiten von Manuskripten
– Besuchen und Auswerten von Fachtagungen, -messen und -ausstellungen
– Zusammenarbeit mit Gutachtern und ggf. selbständiges Testen von der Redaktion zur Veröffentlichung eingereichten Programmen
– Bearbeiten bzw. Beantworten von Leseranfragen.
Falls Sie Interesse an dieser Tätigkeit haben und im Raum Berlin wohnen, rufen Sie uns unter Tel. 2 87 02 03 oder 2 87 03 71 an oder schreiben Sie an:
VEB Verlag Technik
Redaktion MP
Oranienburger Str. 13/14
Berlin
1020

MP

MP-INTERVIEW

INFO '88

Aus Anlaß der INFO '88, die vom 22. bis 26. Februar an der TU Dresden veranstaltet wird, sprachen wir mit dem Vorsitzenden der Gesellschaft für Informatik der DDR (GIDDR), Prof. Dr. Dieter Hammer.

Bei der INFO '88 handelt es sich um den vierten Kongreß der Informatiker der DDR. Welche Ziele hat sich der Kongreß gestellt?
Mit der INFO '88 wird die Reihe der nationalen Konferenzen zur „Entwicklung und Anwendung der elektronischen Rechentechnik der DDR" von 1964 bis 1976 und der INFO-Konferenzen 1977, 1981 und 1984 weitergeführt. Erstmals wird der Informatiker-Kongreß unter Hauptverantwortung der 1985 gegründeten Gesellschaft für Informatik der DDR veranstaltet. Die INFO '88 ist zugleich die wissenschaftliche Hauptkonferenz der Gesellschaft. Vertreter von Einrichtungen der Grundlagenforschung und der Praxis auf dem Gebiet der Informatik und benachbarten Gebieten, wie Automatisierungstechnik und Nachrichtentechnik, sollen neueste Ergebnisse zu Schwerpunkten der Forschung und Applikation vermitteln und diskutieren. Volkswirtschaftliche Aspekte des massenhaften Einsatzes von Informations- und Kommunikationstechnologien gilt es darzustellen und weitere neue Wege aufzuzeigen.

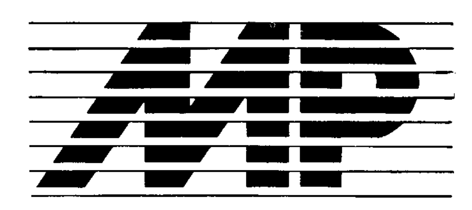
Um solch einen Kongreß durchzuführen, bedarf es sicherlich kompetenter Partner?
Mitveranstalter des Kongresses, bei dem 1400 Teilnehmer erwartet werden – nicht alle Teilnehmerwünsche lassen sich berücksichtigen –, sind die Akademie der Wissenschaften mit dem Institut für Informatik und Rechentechnik und dem Zentralinstitut für Kybernetik und Informationsprozesse, das Hochschulwesen mit dem Informatikzentrum an der Technischen Universität Dresden, die Kombinate Robotron, Datenverarbeitung und Nachrichtenelektronik sowie die Wissenschaftlich-technische Gesellschaft für Meß- und Automatisierungstechnik der Kammer der Technik.

Zu welchen Themen werden Vorträge gehalten, und nach welchen Gesichtspunkten wurden die Beiträge ausgewählt?
Im vergangenen Jahr wurden durch ein Programmkomitee von 24 Fachleuten aus Forschung, Lehre und Praxis aus über 500 eingereichten Vorträgen 245 ausgewählt und zu einem in teilweise 8 Parallelveranstaltungen tagendem Programm in 52 Sitzungen zusammengestellt. Dabei haben wir berücksichtigt, daß jeweils die Hälfte der Teilnehmer aus Forschung und Lehre bzw. aus der Praxis kommen.
Führende Wissenschaftler aus der DDR und anderen sozialistischen Ländern legen in Plenarbeiträgen Trends zu komplexen aktuellen Fragestellungen dar. Das Hauptgeschehen des Kongresses findet natürlich in den Fachsektionen Theoretische Grundlagen, Computertechnik, Software, Künstliche Intelligenz, Komplexe Anwendungen, Aus- und Weiterbildung sowie Gesellschaft und Informatik statt. Außerdem werden während der Tagung Podiumdiskussionen zu „Grundanforderungen an die Informatik für die rechnerintegrierte Fertigung" und zu „Gesellschaftlichen Wirkungen der Informatik" durchgeführt. Parallel zum Vortragsprogramm wird eine Ausstellung Geräte und Softwareprodukte zeigen, die im Ergebnis von Forschung und Entwicklung entstanden sind.

Wir danken Ihnen für das Gespräch und wünschen der INFO'88 einen erfolgreichen Verlauf.

MP

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
1. Jahrgang · ISSN 0232-2892



Mikroprozessortechnik, Heft 12 · 87

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 2 87 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Ingo Paszkowsky, Verantwortlicher Redakteur (Tel.: 2870203); Hans Weiß, Redakteur (Tel.: 2870371); Sekretariat Tel.: 2870381

Gestaltung Christina Kaminski (Tel.: 2870288)
Titel: Christina Kaminski

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

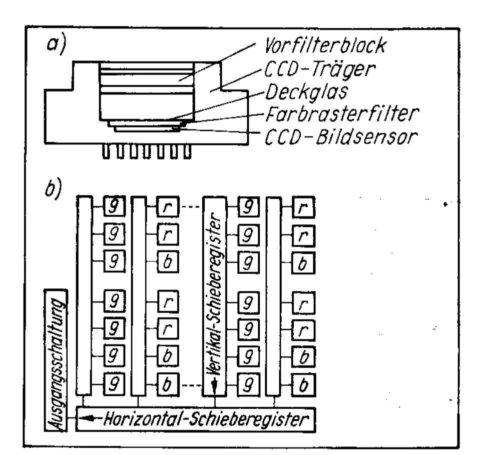
Redaktionsschluß: 20. Oktober 1987

AN (EDV) 49837

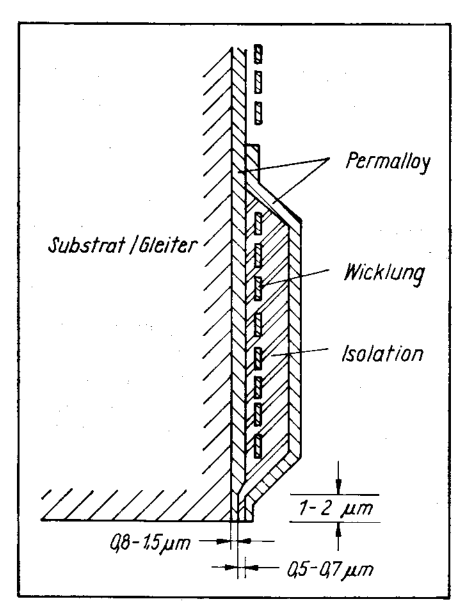
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

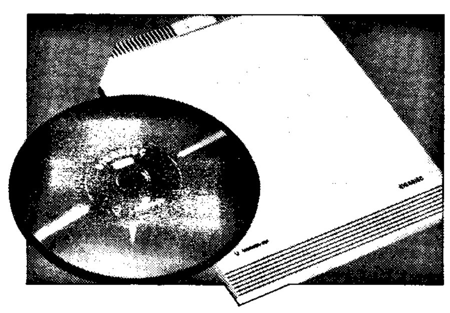
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010, und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Seite 354



Seite 358



Seite 362

Bytes / Wort / Bitfeld: Byte n-1 | Byte n | Byte n+1 | Byte n+2; Byte n-1 | MSB | LSB | Byte n; Byte n-1 | Byte n | 0 ... w-1; Offset; Basisadresse; Basisbit
BCD-Daten gepackt: Byte n-1 | MSD LSD | Byte n+1 | Byte n+2
BCD-Daten ungepackt: Byte n-1 | ... MSD | ... LSD | Byte n

Seite 371

Inhalt

Wirkprinzipien von Informationsaufzeichnungstechnologien

Dr. Walter Kroha, Dr. Harald Baumann
VEB Filmfabrik Wolfen – Fotochemisches Kombinat, Fachdirektorat Forschung und Entwicklung

1. Vorbemerkungen

Mit der raschen Entwicklung der Massenmedien, des Bildungswesens, der Entwicklung aller auditiven, visuellen und audiovisuellen Informations- und Kommunikationsmittel, insbesondere jedoch mit der großen Bedeutung der Schlüsseltechnologien für die Entwicklung der Volkswirtschaft, kommt der schnellen Entwicklung der Informationsaufzeichnungstechnologien eine immer größere Bedeutung zu.
Dabei wird der Tatsache Rechnung getragen, daß die zunehmende Verknüpfung von optischen und akustischen Signalen sowie Computerdaten in vielen Anwendungsgebieten eine die Grenzen des jeweiligen Wirkprinzips überschreitende einheitliche Betrachtung erforderlich macht.
Von entscheidender Bedeutung für die Entwicklung ist es, die Wirkprinzipien der Informationsaufzeichnungstechnologien als Einheit von Informationsaufzeichnung, Informationsspeicherung, Informationsverarbeitung und Informationswiedergabe, also die geschlossene Kette vom Sender zum Empfänger zu betrachten, wobei die komplexen Zusammenhänge zwischen Gerätetechnik und Signalaufzeichnungsmedium berücksichtigt werden müssen.

Das vorliegende Manuskript ist die gekürzte Fassung eines Vortrages, der auf der 7. RGW-Tagung „Magnetische Signalspeicher“ vom 3. bis 8. 5. 1987 in Neubrandenburg gehalten wurde.

Die Speichertechnologie stellt in dieser kompletten Kette ein Schlüsselelement dar. An die Speicher werden hinsichtlich hoher Speicherkapazität, schneller Zugriffszeit, leichter und flexibler Handhabbarkeit, universeller Anwendbarkeit und geringer Kosten hohe Anforderungen gestellt. Speichermedien sind neben den Halbleiterspeichern, welche insbesondere als Arbeitsspeicher Verwendung finden, die Massenspeicher auf Basis Silberhalogenidmaterialien, fotochemische Nichthalogensilbermaterialien, magnetische Materialien und optische Medien. Für die Zukunft werden Technologien auf Basis von holografischen Speichern, Flüssigkristall-, ferroelektrischen und Polymerspeichern und möglicherweise Speicher auf Basis von Molekülen erwartet.

2. Signalaufzeichnungsverfahren für Bild-, Ton- und Computersignale

In Bild 1 sind die wesentlichsten Signalaufzeichnungs- und Speichersysteme für Bild, Ton und Datensignale dargestellt. Nach wie vor kommt der Aufzeichnung von primären optischen Signalen (z. B. Bildsignalen) eine besondere Rolle zu. Die Bildabtastung kann flächenhaft mit einer Kamera bzw. sequentiell mit einem Laserstrahl erfolgen. Der molekulare Sensor übernimmt das Signal, und über Stoffumwandlung, Visualisierung (Entwicklung) und Stabilisierung erfolgt bei fotochemischen Systemen, wie Diazo-Systeme oder Fotopolymer-Systeme, die Signalaufzeichnung und -speicherung /1/.
Die wesentlichsten Innovationen von Bildaufzeichnungssystemen basieren in den letzten Jahren jedoch auf der Verwendung von Fotoleitern als lichtempfindliche Sensoren, die in Form von Dispersionen, Matrizen bzw. als Aufdampf- und Bindemittelschichten eingesetzt werden. Die Absorption von Licht im Fotoleiter erzeugt bewegliche Fotoelektronen im Leitungsband und entsprechende Defektelektronen im Valenzband. Für die Effektivität des bildprägenden Schrittes ist es erforderlich, die Bildung der Elektronen-Loch-Paare irreversibel zu gestalten. Dazu sind folgende Möglichkeiten vorhanden:
① Chemischer Abfang durch Redoxreaktionen an der Fotoleiteroberfläche bzw. im Fotoleiterkristall, wie sie in halbleiterfotografischen Systemen z. B. in der klassischen Silberhalogenidfotografie oder beim Fotodoping realisiert wird.
② Abfang (Neutralisation) durch aufgeprägte elektrische Ladungen z. B. bei elektrofotografischen Systemen.
③ Trennung durch lokale elektrische Felder in Foto- bzw. Halbleitern, die von eingebauten Potentialbarrieren (p/n-Übergänge, Metall-Halbleiter-Übergänge) hervorgerufen werden, z. B. bei elektronischen Bildaufnahmesystemen.
Im Gegensatz zu Variante ① sind bei Variante ② und ③ lichtempfindliche Schicht und Bildspeicher nicht mehr identisch, sondern liegen in örtlich getrennter Form vor. Daraus resultiert die Wiederholbarkeit des Bildprägeschrittes, erfordert jedoch bei den elektronischen Systemen die Zwischenspeicherung der elektrischen Signale auf geeigneten Speichersystemen.
Die Umwandlung von akustischen Signalen in elektrische Signale und deren Speicherung auf magnetische Datenträger ist eine seit vielen Jahren gebräuchliche Technologie im auditiven Bereich und soll nicht näher erläutert werden.
Mit der stürmischen Entwicklung der Computertechnologien in allen Bereichen des Lebens kommt der Speicherung und Archivierung von immer größeren Mengen an Daten eine zentrale Rolle zu. Bis heute sind dazu vor allem magnetische Massenspeicher im Einsatz.

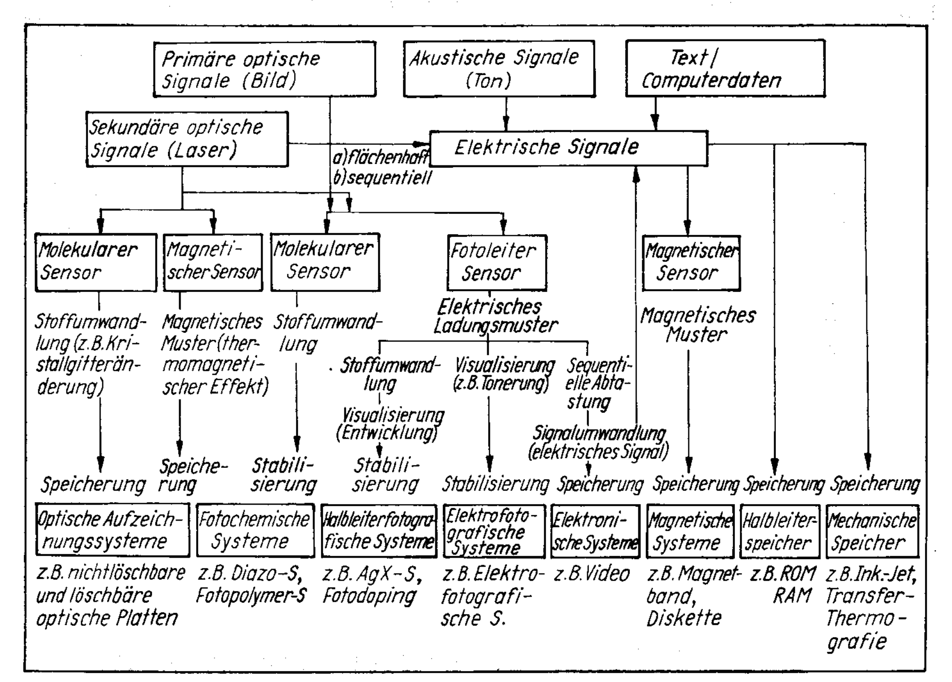


Bild 1 Übersicht ausgewählter Signalaufzeichnungs- und Speichersysteme

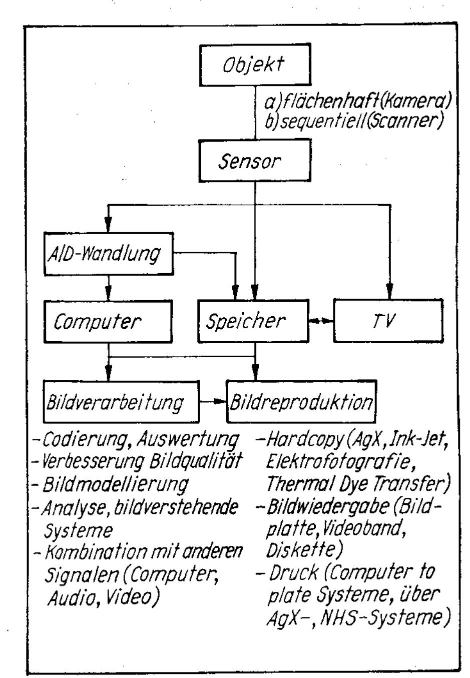


Bild 2 Anordnung Analog/Digital-Wandlung in Bildverarbeitungssystemen

Die Möglichkeit der Übertragung der elektrischen Signale in sekundäre optische Signale, d. h. der Steuerung von Elektronenstrahlen, Lasern oder Halbleiterdioden ermöglicht es, die Signalspeicherung auf völlig neue Aufzeichnungssysteme zu übertragen. Während COM-Systeme (Computer output on microfilm) die analoge Speicherung von Computerdaten auf Silberhalogenid- oder NHS(Nichthalogensilber-)-Materialien bereits mit hoher Effektivität ermöglichen, wobei in zunehmendem Maße die Direktbelichtung der Materialien mit Laserlicht realisiert wird, kommt der Signalspeicherung auf optischen digitalen Datenplatten für die Zukunft eine dominierende Rolle zu. Auch hier werden die bestechenden Eigenschaften des Lasers, seine hohe Fokussierung und seine hohe Energie genutzt, um die Signale nach bestimmten Verfahren zu speichern.
Die Technologien der optischen Datenspeicherung auf NHS-Materialien befinden sich gegenwärtig noch in der Entwicklung, so daß zur Zeit nicht genau gesagt werden kann, welche Systeme sich durchsetzen werden. Die größten Zukunftschancen haben die Systeme, welche auf dem magnetooptischen Effekt beruhen, da dadurch löschbare Datenträger realisierbar sind.
Eine stark zunehmende Bedeutung innerhalb der Signalaufzeichnungs- und Speichersysteme gewinnt die Digitalisierung von Signalen (Bild 2). Sie ermöglicht neben der wesentlichen Erhöhung der Qualität der Speicherung die direkte Verarbeitung durch den Computer bzw. die sofortige Verknüpfung von optischen, akustischen und Computersignalen. Signalspeicherung auf Magnetband oder -platte bzw. optischen Speicher, die enormen Möglichkeiten der Signalverarbeitung, die sich ständig erweiternden Varianten der Signalreproduktion und selbstverständlich die Möglichkeit der Betrachtung auf dem Bildschirm werden bereits in absehbarer Zeit komplexe Systeme der digitalen Video- und Audiokommunikation entstehen lassen.

3. Vergleichende Betrachtung der Leistungsfähigkeit einzelner Informationsaufzeichnungstechnologien

An die Informationsaufzeichnungsmaterialien werden hinsichtlich ihrer Applikation unterschiedliche Anforderungen gestellt. Zur Aufzeichnung primärer optischer Signale werden Materialien benötigt, die den Anforderungen der Kameraempfindlichkeit entsprechen. Das sind vor allem Silberhalogenidmaterialien und elektronische Systeme bzw. bereits stark eingeschränkt Fotopolymere und elektrofotografische Systeme (Bild 3).
Während sich in den vergangenen 150 Jahren die klassische Halogensilberfotografie frei von ernsthafter Konkurrenz durch silberfreie fotografische Systeme entwickeln konnte, erfolgt gegenwärtig die Herausforderung durch die elektronische Fotografie. Insbesondere die rasche Entwicklung der CCD-Matrizen (Bild 4) führte in einigen Anwendungsgebieten zu einer teilweisen Substitution der Silberhalogenidmaterialien durch elektronische Aufzeichnungssysteme /2/.
Der Vergleich der Leistungsfähigkeit von Silberhalogenidschichten und CCD-Sensoren zeigt bei allen Vorteilen der elektronischen

Tafel 1 Vergleich der Leistungsfähigkeit von Silberhalogenidmaterialien und CCD-Sensoren bei ausgewählten Eigenschaften

	Color-Negativ-Kleinbildfilm (24 × 36 mm²), 100 ASA	CCD-Sensor (8 × 8 mm²), magnetische Speicherung auf 2″-Diskette
Sensorelement	Silberhalogenidkristalle Kristall ≠ Pixel	Siliziumkristall (Fotodiode)
Bildelement = Pixel	Silberhalogenidkristalle ca. 250 Mikrokristalle für 64 Halbtonstufen ≙ 1 Pixel	1 Fotodiode ≙ 1 Pixel
Pixelanzahl pro Bild (Durchschnitt)	ca. $3{,}8 \times 10^9$ Kristalle ≙ ca. 15×10^6 Pixel	ca. $0{,}4 \times 10^6$ Pixel/Chip
Informationsspeicher	Sensor = Speicher	Sensor ≠ Speicher magnetische Speicherung (Diskette, Band) optische Speicherung (OD) Festkörperspeicher/Kurzzeit und Einzelbild
Verstärkung	chemische Entwicklung, Verstärkungsfaktor 10^8–10^{10}	elektronische Verstärkung, Verstärkungsfaktor bis 10^4
Bilderzeugung	chemische Änderung des Sensors/Silber- oder Farbstoffbild	· Bildschirmwiedergabe · Hardcopy mittels Printer
Informationskapazität	90×10^6 Bit/Bild	$2{,}4 \times 10^6$ Bit/Bild
Bildzugriffszeit	lang (chem. Entwicklung)	kurz

Tafel 2 Eigenschaften ausgewählter Arbeits- und Massenspeicher

Speichermedium	Speicherdichte (Bit/cm²)	Speicherplätze Kosten (Bit/cent)	Zugriffszeit (s)	Vor-/Nachteile
Arbeitsspeicher				
RAM	10^5–10^6	$5 \cdot 10^0$	10^{-6}–10^{-7}	flüchtig
CCD	10^5–10^6	$1 \cdot 10^1$	10^{-3}–10^{-5}	flüchtig
Magn. Blasenspeicher	10^6	$5 \cdot 10^1$	10^{-2}–10^{-3}	
Massenspeicher				
Magnetplatte	10^4–10^5	$2 \cdot 10^4$	10^{-1}–10^{-2}	
Magnetband	10^6–10^7	$2 \cdot 10^6$	10^0–10^1	
Optomech. Plattenspeicher	10^7–10^8	$2 \cdot 10^7$	10^{-1}–10^{-2}	nicht löschbar
Fotograf. Speicher	10^8–10^{10}	$1 \cdot 10^8$	$> 10^1$	nicht löschbar
Menschliches Gehirn	10^8		1 Bit/s (Langzeit) 50 Bit/s (Kurzzeit)	
Genetische Information	10^{14}		10^3 Bit/s	

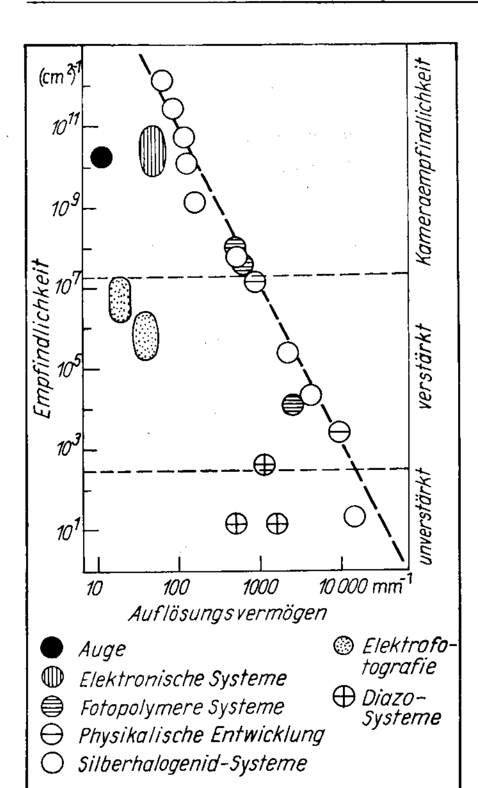
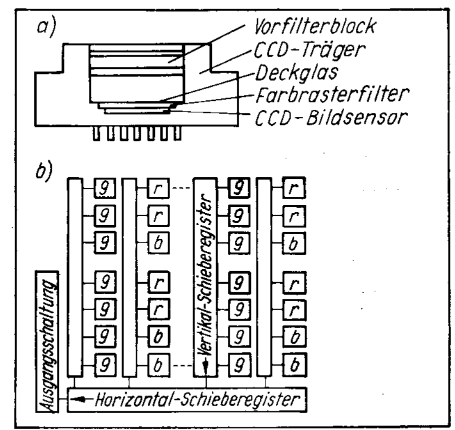


Bild 3 Vergleich des Empfindlichkeits-/Auflösungsvermögens verschiedener Signalaufzeichnungstechnologien

Bild 4 Schematischer Aufbau von CCD-Matrizen



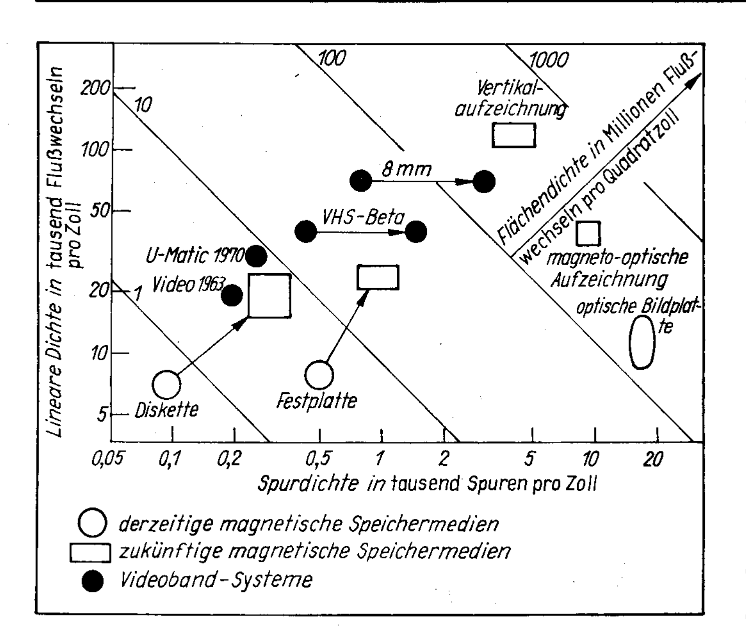


Bild 5 Leistungsfähigkeit gegenwärtiger und zukünftiger Massenspeicher

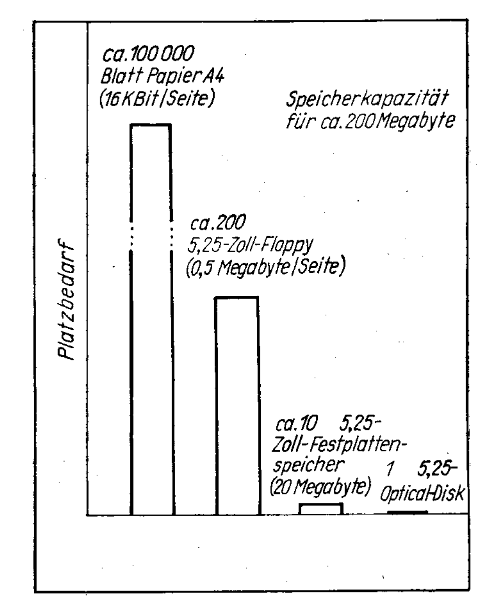


Bild 6 Schematischer Vergleich der Speicherkapazität ausgewählter Massenspeicher

Medien, wie schnelle Zugriffszeit und Wiederverwendbarkeit der Speicher, erhebliche Nachteile hinsichtlich der Empfindlichkeit bzw. des Empfindlichkeitsspektrums und vor allem der Abbildungsqualität (Tafel 1).

So wird sich die Silberhalogenidfotografie auch über das Jahr 2000 hinweg für viele Applikationen weiterhin behaupten, während elektronische Aufzeichnungssysteme die Silberhalogenidmaterialien ergänzen, auf einigen Gebieten verdrängen und neue Anwendungen erzielen.

Die Vorteile der Nichthalogensilbermaterialien liegen bei relativ niedrigen Empfindlichkeiten in den hohen Detailwiedergabeeigenschaften, verbunden mit sehr hohen Informationskapazitäten. Solche Materialien werden deshalb vorwiegend dort eingesetzt, wo genügend Licht vorhanden ist, also in Kopierprozessen oder beim Einsatz leistungsfähiger Lichtquellen wie Elektronenstrahlen oder Laser. Sie haben als „Spezialisten" auf bestimmten Anwendungsgebieten wie z. B. der Herstellung von Druckplatten, in COM-Systemen, in Laserdruckern bzw. in der Mikrolithografie eine hohe Effizienz erreicht und zu optimalen Applikationen geführt. Bei relativ geringen Speicherkosten muß jedoch die Nichtlöschbarkeit und die relativ hohe Zugriffszeit für die gespeicherte Information berücksichtigt werden.

Zur direkten Aufzeichnung elektrischer Signale, seien es Computerdaten oder transformierte akustische bzw. optische Signale, werden zwei Arten von Datenspeichern eingesetzt (Tafel 2)/3/:

a) Arbeitsspeicher mit mäßig hohen Datenvolumina (derzeit $\leqq 1{,}0 \cdot 10^6$ Bit je Einzelspeicher) und extrem kurzen Zugriffszeiten ($< 10^{-6}$ s). Zukünftig sind Arbeitsspeicher von 4 ... 16 Megabit zu erwarten.

b) Massenspeicher mit hohen Datenvolumina ($> 10^6$ Bit je Einzelspeicher und mittleren Zugriffszeiten ($> 10^{-4}$ s).

Die wichtigsten Massenspeicher sind derzeit die *magnetomotorischen Speicher*. Ihre Leistungsfähigkeit wird durch die realisierbare Anzahl von magnetischen Flußwechseln in der Spur (lineare Speicherdichte, Flußwechsel/mm, Bit/mm und die pro mm Speicherbreite anordenbare Anzahl von magnetischen Spuren) bestimmt. Daraus ergibt sich als wichtigstes Leistungskriterium die Flächenspeicherdichte (Flw/mm²). Bild 5 zeigt einen interessanten Vergleich gegenwärtiger und zukünftiger Speichermedien für die verschiedensten Applikationsgebiete.

Dr. rer. nat. Walter Kroha, Jahrgang 1950, absolvierte von 1971 bis 1975 ein Chemiestudium an der Technischen Hochschule „Carl Schorlemmer" Leuna-Merseburg. Seit 1975 ist er im Fachdirektorat für Forschung und Entwicklung in den Bereichen Grundlagenforschung, Synthese und Schwarzweiß-Materialien beschäftigt. 1981 Promotion A auf dem Gebiet der Dispergierung hydrophober Bauelemente. Seit 1987 ist Walter Kroha Bereichsdirektor für Erzeugnisentwicklung des VEB Filmfabrik Wolfen.

Dr. sc. nat. Harald Baumann, Jahrgang 1951, studierte Chemie von 1970 bis 1974 an der Technischen Hochschule „Carl Schorlemmer" Leuna-Merseburg. An der gleichen Hochschule war er ab 1974 wissenschaftlicher Assistent, ab 1979 Oberassistent. 1979 erfolgte Promotion A und 1986 Promotion B zu Fragen der Fotochemie. 1983/84 Zusatzstudium an der Moskauer Staatlichen Lomonossow-Universität. Seit 1984 ist Harald Baumann beschäftigt im Fachdirektorat für Forschung und Entwicklung des VEB Filmfabrik Wolfen als Leiter der Hauptabteilung Nichthalogensilbermaterialen.

Gegenwärtig dominiert noch die Aufzeichnung mit Magnetköpfen auf anisotropen magnetischen Medien für die Längsspeichertechnik, unter Nutzung von magnetischen Pigmenten eingebettet in einer Bindemittelmatrix. Die Tendenz geht zu immer kleineren Teilchengrößen der Pigmente (bei exakt definierter Teilchengeometrie) und höheren Koerzitivfeldstärken (Tafel 3).

Tafel 3 Ausgewählte Eigenschaften magnetischer Pigmente

Magnetisches Pigment	Kantenlänge [μm]	Koerzitivfeldstärke [A/cm]
Fe_2O_3	0,5	280–320
Co modif. Fe_2O_3	0,4	400–640
CrO_2	0,4	400–540
Fe	0,25	800 (1250)
Bariumferrit	0,08 × 0,03	600–1400
Metalldünnschicht	ca. 100 mm Schichtdicke	800–1400

Die mit den Methoden der *Dünnschicht-Technologie* hergestellten Speichermedien besitzen gegenüber den in Antragstechnik hergestellten Partikel-Polymer-Schichten einige Vorteile, wie z. B. Speicherdichten bis 10 Megabit/cm²; die technologische Herstellung und die Anwendung dieser Medien ist jedoch gegenwärtig noch sehr problematisch, so daß die Pigmentsysteme hinsichtlich ihrer Leistungsfähigkeit weiter entwickelt werden (z. B. Bariumferrit-Speicher). Die weitere Erhöhung der Speicherdichte von 20 Megabit/cm² und darüber hinaus soll mit sogenannten isotropen, das heißt richtungsunabhängigen, zum Beispiel kobaltdotierten Eisenoxidpartikeln, erreichbar sein.

Der Übergang zur *Vertikalspeicherung* (Bild 5) ermöglicht eine weitere Erhöhung der Speicherdichte. Die Vorteile der Vertikalaufzeichnung bestehen in der wesentlich verringerten Gefahr der Selbstentmagnetisierung und der erhöhten Packungsdichte der Magnetpigmente durch ihre Ausrichtung vertikal zur Oberfläche des Trägermaterials. Bei prinzipieller Beibehaltung der Beschichtungstechnologie und der Schreib-/Lese-Köpfe sind Speicherdichten bis 40 Megabit je cm² denkbar.

Der Einsatz der magnetischen Materialien kann entsprechend den unterschiedlichen Anwendungsgebieten als Festplatte, Floppy, Datenband oder Videoband erfolgen.

Die stürmische Entwicklung der Lasersysteme ermöglicht eine neue Speichertechnologie, die sogenannte *optische Speicherung* /4, 5, 6, 7/. Die Hauptvorteile der optischen Datenspeicherung gegenüber der magnetischen ergeben sich aus der Verwendbarkeit von Laserstrahlen zum Aufzeichnen und Auslesen der Informationen und damit um den Faktor 10 höhere Spurdichte (Bild 6) und ein berührungsloses Aufzeichnen und Auslesen der Informationen. Die gegenwärtigen optischen Speicher kann man wie folgt klassifizieren:

■ Nichtlöschbare optische Nur-Lese-Speicher (CD-ROM – Compact Disc–Read Only Memory). Die Herstellung einer Masterplatte mittels Laserverfahren und die Produktion einer größeren Anzahl von Duplikaten nach bestimmten Verfahren (z. B. Prägen) erfolgt beim Plattenhersteller /8/.

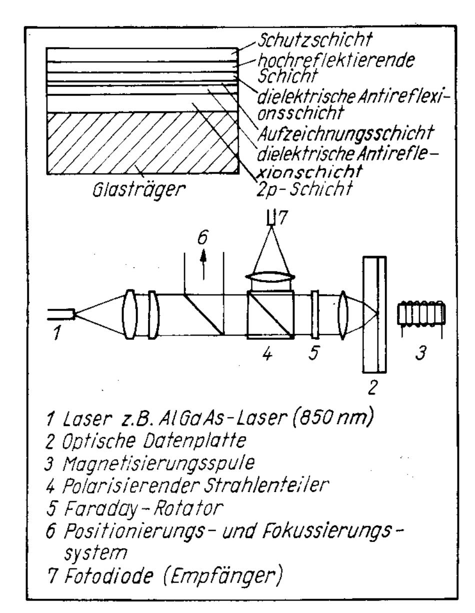


Bild 7 Schematischer Aufbau einer optomagnetischen Datenplatte und prinzipielle Funktionsweise der Datenein- und Ausgabe

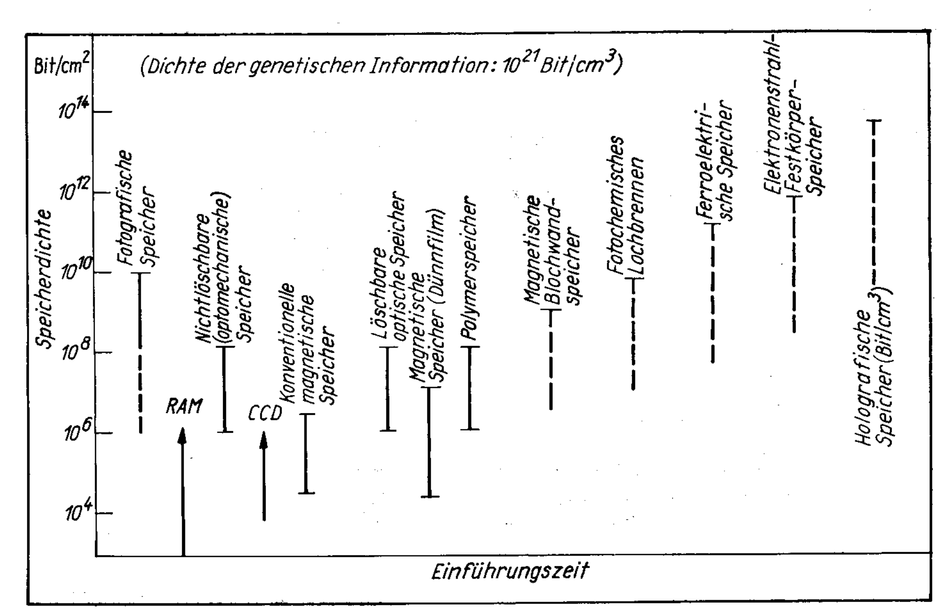


Bild 8 Entwicklungstendenzen bei Massenspeichern

■ Nichtlöschbare optische Speicher (DRAW – Direct Read After Write bzw. O-ROM – Optical Read Only Memory). Die Dateneingabe erfolgt durch den Nutzer. Diese einmal beschreibbaren Speicher nutzen die thermische Einwirkung eines fokussierten Laserstrahls (ca. 10 mW, Impulslänge ca. 50 ns) zur Schichtveränderung. Das Auslesen erfolgt mit einem Laser geringerer Intensität (ca. 0,6 mW) unter Ausnutzung des geänderten Reflexions- bzw. Lichtstreuverhaltens der Löcher, Blasen oder Vertiefungen.

■ Die größten Zukunftschancen haben die löschbaren optischen Speicher (E-DRAW – Erasable DRAW) mit ca. 200–500 Megabit/cm² (zum Vergleich besitzt ein Megabit-Halbleiterspeicher eine Speicherdichte von maximal 2 Megabit/cm²).

Hier zeichnen sich folgende Entwicklungsrichtungen ab:

● Magneto-optische Speicher, bestehend aus dünnen Schichten mit reversibel umkehrbaren Magnetisierungsrichtungen, die beim Auslesen die durch den Kerr- oder Faraday-Effekt bewirkte Drehung der Polarisationsebene des Lichtes nutzen /9/ (Bild 7).

● Optische Speicher, die auf reversiblen Phasenübergängen amorph/kristallin metallischer bzw. metalloxidischer Schichten basieren.

● Optische Speicher, welche die durch Laser bewirkte reversible Farb- und/oder Reflexionsänderung ausnutzen.

Die Lösung der vielfältigen Probleme der optischen Datenplatten (Speicherschicht, Gerät, Computerschnittstellen usw.) könnte bereits in absehbarer Zeit zu einem Durchbruch dieser Innovation mit breiter Massenanwendung führen.

Für die weitere Zukunft sind Speicher mit Dichten über 10^8 Bit/cm² (Bild 8), z. B. magnetische Blochwand-Speicher bis 10^9 Bit/cm² oder Elektronenstrahl-Festkörper-Speicher bis 10^{12} Bit/cm² sowie Systeme mit 3dimensionaler Speichermöglichkeit (z. B. holografische Speicher) mit theoretischen Speicherdichten von 10^{14} Bit/cm³ zu erwarten /3, 7/.

Damit erreichen diese Speicher Werte der klassischen optischen Speicher in Form von Silberhalogenidschichten von 10^7 bis 10^8 Bit/cm² beziehungsweise Diazomaterialien von 10^{10} Bit/cm².

4. Signalaufzeichnungsverfahren und Anwenderbedürfnisse

Die unterschiedlichsten Anwenderbedürfnisse für Bild-, Ton- und Datenspeicherung haben zu den verschiedensten Applikationsfeldern der beschriebenen Informationsaufzeichnungsmaterialien geführt, denn keines der Informationsaufzeichnungssysteme ist in der Lage, alle Anwenderforderungen gleichermaßen zu erfüllen. Überwiegend sind in den einzelnen Applikationsgebieten mehrere Aufzeichnungssysteme vertreten, welche häufig in Konkurrenz zueinander stehen. Tritt nun auf einem Applikationsfeld ein durch die Anwendung neuer Wirkprinzipien gekennzeichnetes Signalaufzeichnungssystem auf, wird sofort die Frage nach der Verdrängung der konventionellen Systeme gestellt und dies häufig sehr kontrovers diskutiert. Das war Ende der 60er, Anfang der 70er Jahre beim Auftreten neuer silberfreier fotochemischer Informationsaufzeichnungsmaterialien der Fall, wird gegenwärtig mit der Darstellung neuer elektronischer Stehbildkameras erneut diskutiert und ist bei der Diskussion der Datenspeicherung auf Band oder Diskette oder optischer Datenplatte aktuell. Charakteristisch für den Sektor Informationsaufzeichnungstechnologien ist jedoch in den meisten Fällen ein Nebeneinander konventioneller und neuer Technologien, und nur in wenigen Fällen (z. B. Ersatz des Silberhalogenidmaterials durch die Xerografie auf dem Büroko-

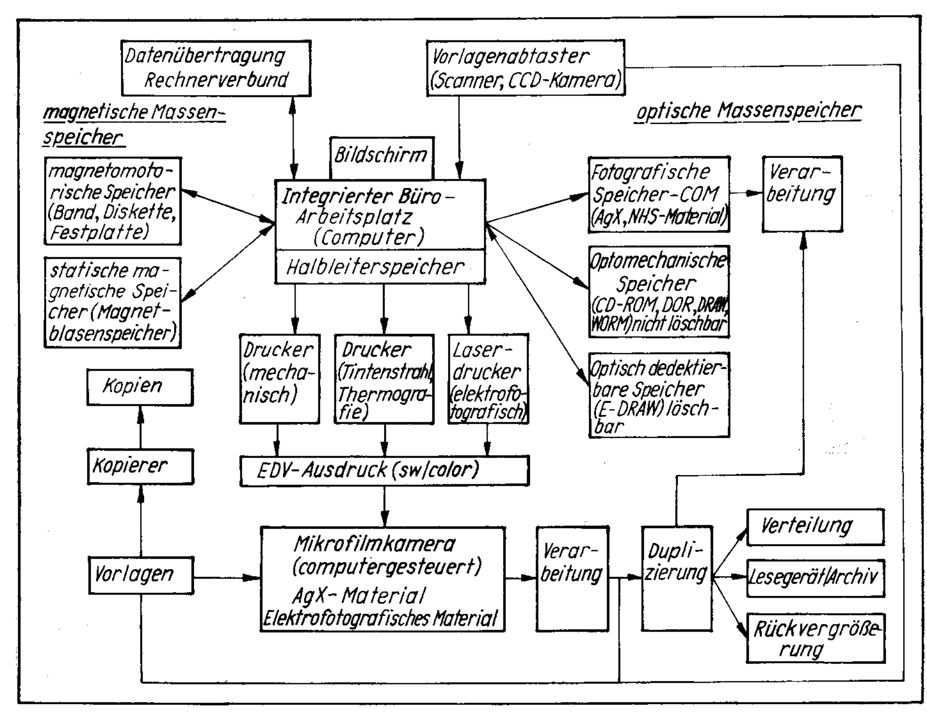


Bild 9 Hybridsysteme bei integrierten Büro-Arbeitsplatz-Computern

piersektor) erfolgte eine vollständige Verdrängung konventioneller Systeme (das hat natürlich nichts mit Sortimentsbereinigungen wie z. B. auf dem Amateurvideogebiet zu tun).
Neben dem parallelen Einsatz konventioneller und neuer Informationsaufzeichnungssysteme und deren rascher Entwicklung zu ausgereiften Technologien bieten sich interessante Lösungen an, welche die Vorteile der konkurrierenden Informationsaufzeichnungstechnologien unter Nutzung von Teilelementen beider Technologien verbinden, um so zu neuen Qualitäten zu kommen. Derartige Hybridsysteme, wie sie auch bei verschiedenen anderen technischen Systemen bekannt sind (wie z. B. Verbrennungs- und Elektromotoren als Hybridsysteme in Kraftfahrzeugen, Einsatz monolithisch integrierter Schaltkreise und diskreter Bauelemente in Hybridschaltkreisen, hybridfaseroptische Sensoren mit elektrischem Aufnehmer und optischer Übertragung) existieren bereits seit einigen Jahren auf dem Sektor der Informationsaufzeichnungstechnologien, und ihre Bedeutung wächst ständig/10/.
Auch bei integrierten Arbeitsplatz-Computern (Bild 9), welche zur Rationalisierung von Verwaltungsarbeiten, der Informationsverarbeitung, in Konstruktionsbüros oder zur Steuerung von Produktionsanlagen in verschiedensten Gerätekonfigurationen eingesetzt werden, setzen sich zunehmend Hybridsysteme durch. Neben den in den Rechnern integrierten Halbleiterspeichern kommt der Massenspeicherung von Daten eine dominierende Rolle zu: Die Massenspeicher weisen im Vergleich zu den Halbleiterspeichern wesentlich geringere Kosten je Speicherplatz auf und können zudem die Daten ohne Energiezufuhr über lange Zeiträume erhalten. Die externen Speicher lassen sich im Prinzip uneingeschränkt erweitern und bei Großrechnern um einige 1000mal größer gestalten, als die interne Kapazität von einem bis mehreren Megabyte.
Die Basis der Massenspeicherung maschinenlesbarer Daten und Informationen bildet bis heute die magnetomotorische Aufzeichnung auf Bänder, Disketten und Platten /11, 12, 13, 14/, wobei letztere das Spitzenniveau von 4 Megabit/cm^2 erreichen. Der Einsatz reicht von 14-Zoll-Standgeräten mit einer Kapazität von 0,2 ... 5 Gigabyte für Großrechner bis zu 3,5-Zoll-Geräten mit Kapazitäten von 6 ... 50 Megabyte für Arbeitsplatzcomputer. 8-Zoll-Festplattenlaufwerke ermöglichen z. Z. in Verbindung mit modernsten Mikroprozessoren die Realisierung der 16-Bit- und 32-Bit-Kompaktrechner.
Trotz der zunehmenden Bedeutung von Festplattenspeichern haben auch Diskettenspeichergeräte in absehbarer Zeit ihre Existenzberechtigung, wobei hier die Entwicklung von 5¹/₄-Zoll- auf 3,5-Zoll-Technologie bei Erzielung der Kapazitäten von 8-Zoll-Disketten von ca. 1,6 Megabyte bevorsteht.
Aber auch die klassische Magnetspeichertechnologie ist nach wie vor zur Datensicherung und -archivierung in sogenannten back-up-Systemen vorerst unerläßlich.
Statische magnetische Speicher wie Magnetblasenspeicher behaupten bisher keine starke Marktposition, könnten jedoch bei Speicherkapazitäten von ca. 64 Megabit gegen Ende des Jahrzehnts an Bedeutung gewinnen.
Zunehmende Bedeutung erlangen jedoch die optischen Massenspeicher infolge ihrer hohen Speicherkapazität. Während die COM-Systeme einen festen Platz insbesondere als Langzeitarchivmedium einnehmen und hinsichtlich der Speicherkosten den magnetischen Speichern etwas, den derzeit vorliegenden optischen Speicherplatten stark überlegen sind, werden sich optische Speicherplatten insbesondere dort durchsetzen, wo zu ergänzende und zu aktualisierende Informationen gespeichert werden sollen bzw. mehrere Anwender zur gleichen Zeit zugreifen müssen. Interessante Anwendungsgebiete ergeben sich ebenfalls in CAD/CAM-Systemen, bei der Langzeitspeicherung von EDV-Daten und nicht zuletzt in allen beschriebenen Bildbearbeitungssystemen. Die neuen Möglichkeiten der optischen Informationssysteme mit Bilderfassung über Dokumenten-Scanner, Kameras, Filmscanner, Wiedergabe auf hochauflösenden Bildschirmen und Faksimiledruckern, Verteilung über leistungsfähige optische Netze (Glasfaserkabel), Anschluß von Text-, Satz- und Faxsystemen und im Zentrum der Speicherung auf optischen Platten, Silberhalogenid- oder NHS-Materialien bzw. magnetischen Speichern gilt es, die ganze Aufmerksamkeit zu schenken.

Literatur

/1/ Silberfreie Bildaufzeichnungssysteme – Ergänzung oder Alternative?/Böttcher, H.; Görgens, E.: Chem. Techn. 37 (1985) 137
/2/ Der CCD-Chip im Detail/Liebert, H. P.: Funkschau, (München) 18 (1986) 46
/3/ Polymere als Träger und Speicher von Informationen/Kämpf, G.: Ber. Bunsen Ges. Phys. Chem. 89 (1985) 1179
/4/ Optische Speicherplattensysteme: Die Kosten-Nutzenbarriere ist noch sehr hoch/Bauernfeind, U.: Mikrodok 5 (1986) 156
/5/ Optische Datenspeicher auf plattenförmigen Speichermedien/Bergmann, H.: Bild und Ton 39 (1986) 19
/6/ Digital-optische Verfahren zur Datenspeicherung/Kusch, S.; Bohm, J.: Bild und Ton 40 (1987) 79
/7/ Digitale Speichermedien der Zukunft/Engemann, J.: Fernseh- und Kinotechnik 41 (1987) 33
/8/ Zur Einordnung der Compact-Disc/Hohmuth, G.: Bild und Ton 40 (1987) 28
/9/ Signalaufzeichnung in magnetooptischen Schichten: Bohm, J.: Kusch, S.: J. Inf. Rec. Mater. 14 (1986) 235
/10/ Hybridlösungen bei Informationsaufzeichnungssystemen (I)/Kroha, W.; Witt, W.: Bild und Ton 40 (1987) 337
/11/ Über die zukünftige Entwicklung magnetomotorischer Speichermedien/Stopperka, K.: Bild und Ton 40 (1987) 19
/12/ Eigenschaften und Möglichkeiten magnetomotorischer Speichermedien/Siakou, M.: J. Signal AM 9 (1981) 345
/13/ Zur Perspektive magnetomotorischer Speicher/Völz, H.: J. Signal AM 9 (1981) 333
/14/ Plattenspeicher – Stand und Entwicklungstendenzen/Salzmann, G.: J. Inf. Rec. Mater. 14 (1986) 245

✉ KONTAKT ☏
VEB Filmfabrik Wolfen
Dr. W. Kroha, Puschkinstraße 1, Wolfen, 4440

Plattenspeicher

Stand und Entwicklungstendenzen

Dr. Günter Salzmann
VEB Robotron-Elektronik Dresden,
Direktionsbereich Wissenschaft und Technik

1. Magnetomotorische Plattenspeicher

1.1. Systemtechnische Aspekte

Nach wie vor wird die Leistungsfähigkeit elektronischer Daten- und Informationsverarbeitungssysteme in starkem Maße von der Ausstattung mit Speichern der ersten Peripherie bestimmt. Dafür sind Speicher großer Kapazität mit wahlfreiem Blockzugriff und kurzen Zugriffszeiten erforderlich, die eine hohe Zuverlässigkeit, geringe Fehlerrate und einen günstigen Preis aufweisen. Diese Forderungen lassen sich bis heute am besten mit magnetomotorischen Plattenspeichern erfüllen, wobei sich der Festplattenspeicher in Winchestertechnologie weitestgehend durchgesetzt hat. Da das Platte-Kopf-System in Reinraumatmosphäre betrieben wird, konnten die Speicherdichte und eine Reihe weiterer Leistungsparameter ständig verbessert werden.
Die Entwicklung vollzieht sich außerordentlich schnell, nicht zuletzt durch die Besonderheit begünstigt, daß das Speichermedium nicht austauschbar ist. Damit entsteht ein relativ großer Freiraum für Neuentwicklungen ohne Einschränkungen durch offene Standardisierungsfragen oder Begrenzungen durch kritische Passungen. Es genügt, bestimmte Kompatibilitätsforderungen einzuhalten, was keine besonderen Schwierigkeiten bereitet.

1.2. Platte-Kopf-Paarung

1.2.1. Entwicklung der Speicherdichte

Wesentlichste Grundlage des technischen und ökonomischen Fortschritts bei Plattenspeichern ist die Erhöhung der Speicherdichte. Aus der Entwicklung der letzten Jahrzehnte ist abzuleiten, daß die Flächenspeicherdichte in 10 Jahren um eine Größenordnung zunimmt, was einer Verdopplung in 2 bis 3 Jahren entspricht /1/, /2/. In den Jahren 1984–1986 lagen die Höchstwerte bei 34000 ... 38000 Bit/mm^2, repräsentiert beispielsweise durch die 5¹/₄"-Geräteserie EXT 4000

von Maxtor und Megafile von Siemens sowie die 2,6-MByte-Spindel 3380E von IBM auf der Basis von 14"-Platten. Mit Erscheinen der 5¹/₄"-Baureihe EXT 8000 von Maxtor, bei welcher die Speicherdichte 67000 Bit/mm² beträgt, bestätigte sich diese Regel Anfang 1987 erneut.

Das für eine Speicherdichte > 3000 Bit/mm² erforderliche technologische Niveau wird wesentlich durch den Einsatz von Metalldünnschichtplatte und teilweise Dünnschicht-Magnetkopf geprägt. Aber auch verbesserte Signalverarbeitungsverfahren und Servosysteme für die Spurpositionierung sowie höhere mechanische Präzision leisten dazu einen Beitrag.

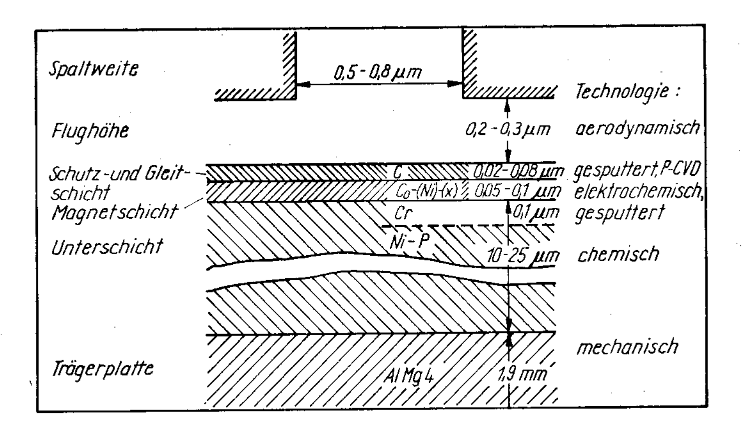


Bild 1 Kopf-Platte-Paarung mit Metall-Dünnschichtmedium für Speicherdichten bis etwa 40000 Bit/mm²

1.2.2. Fortschritte bei der Plattentechnologie

Gegenüber der Partikelschicht auf Basis γ-Fe^2O_3 hat die metallische Speicherschicht den Vorteil einer wesentlich höheren Sättigungsmagnetisierung, des geringeren effektiven Arbeitsabstandes, das heißt des Abstandes Kopfspiegel bis Mitte der Speicherschicht bei gegebener Flughöhe des Magnetkopfes, eines größeren Wiedergabesignals und eines kleineren Rauschpegels, woraus eine höhere Speicherdichte resultiert /3/. Die Grundparameter einer typischen Platte-Kopf-Paarung für das Niveau 30000 ... 40000 Bit/mm² veranschaulicht Bild 1. Auf der AlMg-Trägerplatte wird eine Ni-P-Schicht chemisch niedergeschlagen. Sie dient zur Verbesserung der Härte, Polierfähigkeit und chemischen Beständigkeit sowie zur Isolation von Fehlstellen der Trägerplatte. Da die zu erreichenden magnetostatischen Parameter der Speicherschicht eine definierte Textur der Unterlage erfordern, ist im allgemeinen auf der polierten Ni-P-Unterschicht eine weitere Zwischenschicht notwendig. Dafür hat sich Chrom als geeignet erwiesen. Als Speicherschicht wird üblicherweise eine Zwei- oder Dreikomponentenlegierung mit Cobalt als Hauptbestandteil und H_c = 450 bis 700 A/cm eingesetzt.

Entscheidend für die Betriebssicherheit und Lebensdauer des Speichermoduls ist eine geeignete Schutz- und Gleitschicht. Sie muß einen geringen Reibungskoeffizienten aufweisen, mindestens 10000 Start-Stopp-Zyklen des Magnetkopfes ohne nennenswerten Verschleiß überstehen und ausreichenden Korrosionsschutz bieten. Hierfür hat sich in den letzten Jahren Kohlenstoff als günstiges Material erwiesen, der im Sputter- oder Plasma-CVD-Verfahren aufgebracht wird und dessen Eigenschaften durch das technologische Regime in weiten Grenzen einstellbar sind /4/.

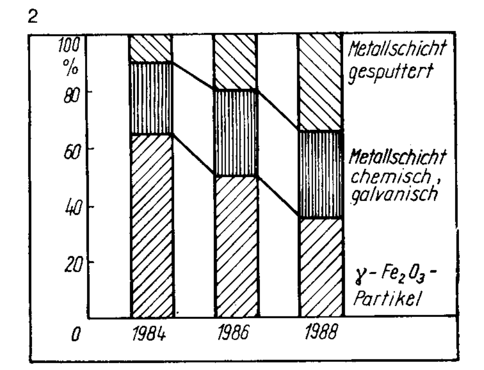
Neben den speichertechnisch relevanten Parametern der Platte-Kopf-Paarung sind bei einer Flughöhe von 0,2 ... 0,3 μm sehr hohe Anforderungen an die mechanische Oberflächengüte und Fehlerfreiheit der Platte zu stellen /5/. Die maximale Rauhigkeit und singuläre Oberflächenerhebungen dürfen über den gesamten Nutzbereich der Oberfläche ca. 0,1 μm nicht überschreiten. Metalldünnschichtplatten werden seit 1983/1984 in größeren Stückzahlen vor allem für 5¹/₄"-Laufwerke im oberen Leistungsbereich produziert. Die Speicherschicht wurde bisher vorwiegend chemisch oder galvanochemisch niedergeschlagen. Zunehmend findet die Sputtertechnologie Anwendung, von der bessere Steuerbarkeit und Flexibilität der technologischen Prozeßschritte sowie höhere Ausbeute erwartet werden. Trendeinschätzungen besagen /6/, daß 1988 die Sputtertechnologie bereits vorherrschend sein und bezüglich der produzierten Plattenstückzahl die γ-Fe_2O_3-Partikelschichten überholt haben wird (Bild 2).

Eine bemerkenswerte Neuentwicklung auf dem Gebiet der Plattentechnologie ist die SSR-Platte (Stretched Surface Recording). Träger des Schichtsystems ist eine Folie, die im Abstand von 0,3 μm über einen festen Plattenkern gespannt wird. Diese zunächst für das 5¹/₄"-Format verfügbare Platte läßt sich relativ billig herstellen und bietet eine höhere Sicherheit bezüglich der durch Schmutzpartikel entstehenden Gefahr von Kopf-Platte-Havarien. Die SSR-Platte läßt sich sowohl für Winchesterspeicher wie auch als Wechselkassette einsetzen, ähnlich einer Diskette erhöhter Kapazität.

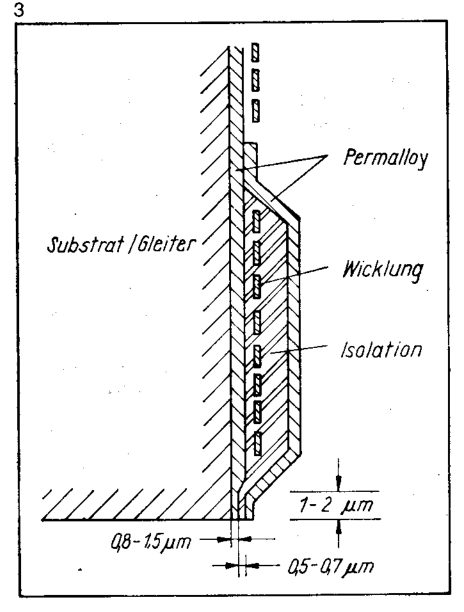
1.2.3. Dünnschichtmagnetkopf

Wachsende Bedeutung für die hochdichte Speicherung wird der Dünnschichtmagnetkopf erlangen. Auf Grund des noch hohen Preises und der beschränkten Verfügbarkeit wird er gegenwärtig nur in Hochleistungsgeräten eingesetzt. Gemäß Bild 3 werden die Magnetschenkel aus hochpermeablen NiFe-Schichten gebildet. Die technischen Vorteile dieses Kopftyps im Vergleich zum Ferritkopf liegen in der höheren Sättigungsmagnetisierung, in einer geringeren Induktivität und einem guten Wirkungsgrad bei kleinen Spaltweiten. Im Herstellungsprozeß sind Elemente der mikroelektronischen Technologien nutzbar, insbesondere für die Strukturierung. Die Forderung nach einer großen inneren Spaltweite im Bereich der Kopfwicklung führt funktionell bedingt zu einem dreidimensionalen Gebilde. Anders als in der Mikroelektronik sind auch die Materialien, die Schichtdicken und das Substrat. Wie Bild 4

2



3



4

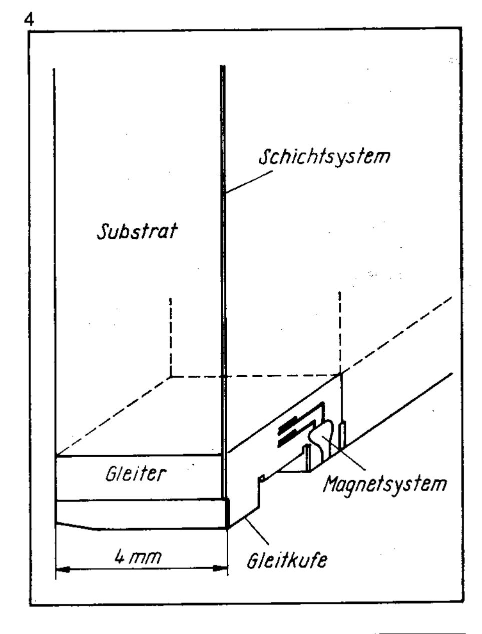


Bild 2 Entwicklung der Magnetschichtentechnologien bei Winchesterplatten

Bild 3 Prinzipieller Aufbau eines Winchester-Dünnschichtkopfes (Querschnitt)

Bild 4 Gleitertechnologie beim Winchester-Dünnschichtkopf

veranschaulicht, wird aus dem Substrat der Gleitkörper herausgearbeitet /7/. Das erfordert ein Substrat von 4 mm Dicke und spezielle Materialeigenschaften, die den Belastungen durch die mechanische Bearbeitung und durch die Start-Stopp-Vorgänge gewachsen sind.

1.3. Weiterentwicklung der Gerätetechnik

1.3.1. Winchesterlaufwerke und Speichergeräte

Die Entwicklung der Plattenlaufwerke vollzieht sich in bestimmten Schritten, die sich nach dem zu einem gegebenen Zeitpunkt beherrschten technologischen Niveau richten, und im Rahmen bestimmter Geräteklassen, die sich nach system- und anwendungstechnischen Erfordernissen herausgebildet haben. Die gebräuchlichen Geräteklassen und den gegenwärtigen technischen Stand charakterisiert Tafel 1. Der Fortschritt bei der Beherrschung größerer Speicherdichten wird genutzt, um die Speicherkapazität innerhalb der jeweiligen Geräteklasse bei verringerten relativen Speicherkosten zu erhöhen oder die Laufwerkabmessungen zu verkleinern. Das führt international zu einer ständigen Verschiebung der Kapazitätsbereiche in den Geräteklassen und deren Anwendungsgebieten. Der erreichte Stand bei den 8″-Laufwerken ermöglicht, damit die 14″-Geräte bis 330 MByte vollständig abzulösen. Auch die 5¹/₄″-Laufwerke reichen bereits in diesen Kapazitätsbereich hinein. Sie sind in Einzelfällen auch bezüglich der für Mehrnutzersysteme und Multitaskverarbeitung wichtigen Zugriffszeit konkurrenzfähig. Als Alternative zu den 14″-Geräten haben im oberen Kapazitätsbereich auch Laufwerke mit 9″ und 10,5″ Plattendurchmesser in den letzten Jahren Bedeutung erlangt.
In der DDR wurden vom VEB Kombinat Robotron die Festplattenspeicher K 5501 mit 39 MByte Speicherkapazität und K 5502 mit 160 MByte (unformatiert) entwickelt. Die 4 Speicherplatten verwenden ein metallisches Dünnschicht-Speichermedium, das auf elektrochemischem Wege hergestellt wird. Die Geräte sind in Einschubbauweise mit dem Standardinterface SMD ausgeführt und vorwiegend für den Einsatz in Kleinrechnern vorgesehen.

1.3.2. 5¹/₄″-Laufwerke

Nach wie vor ist eine außerordentlich hohe Entwicklungsdynamik in der Klasse der 5¹/₄″-Laufwerke zu verzeichnen. Der Rahmen ist durch folgende Industriestandards abgesteckt:

Günter Salzmann ist im Kombinat Robotron Dresden an Forschungsaufgaben zu magnetischen Plattenspeichern tätig. Nach seinem Studium an der Technischen Universität Dresden, Fakultät für Elektrotechnik, promovierte er auf dem Gebiet der Mikrowellentechnik. Er war an Forschungsaufgaben zu magnetischen Dünnschicht-Operativspeichern, zur optischen Zeichen- und Informationsverarbeitung sowie zur Klein- und Mikrorechentechnik beteiligt.

Brutto-Spurkapazität: k × 10416 Byte
Netto-Spurkapazität: k × 8192 Byte
Datenübertragungsrate: k × 5 MBit/s
Interface:
ST 506/412 für k = 1
ESDI oder SCSI für k ≧ 2
Abmessungen:
5¹/₄″-Formfaktor mit 83 mm × 146 mm × 203 mm oder slim-line mit 41 mm Höhe.

Die Mehrzahl der Laufwerke wird gegenwärtig mit k = 1 für Kapazitäten von 20–40 MByte und in slim-line-Ausführung hergestellt. Sie werden vor allem in Personalcomputern eingesetzt. Überwiegend kommen γ-Fe_2O_3-Partikelschichten und Ferritköpfe zur Anwendung. Für gehobene Leistungsansprüche sind Laufwerke um 80 MByte mit k = 1 und k = 2 üblich, wobei im letzteren Fall hauptsächlich Metallschichtplatten eingebaut werden. Als nächster Konzentrationspunkt zeichnet sich der Kapazitätsbereich um 170 MByte mit k = 2 ab.
Als erstes Erzeugnis im Niveau k = 3 brachte die Firma Maxtor Anfang 1987 die Serie EXT 8000 heraus /8/. Mit 8 Platten werden eine Kapazität von 760 MByte und eine mittlere Positionierzeit von nur 18 ms realisiert. Bei einer Längsspeicherdichte von 1240 Bit/mm und 54 Spuren/mm beträgt der gegenwärtige Spitzenwert der Flächenspeicherdichte 67 000 Bit/mm².

1.3.3. 3,5″-Laufwerke

Die Klasse der 3,5″-Laufwerke hat sich zu einem Entwicklungsschwerpunkt herausgebildet. Da bezüglich des Datenformats und des Interfaces aus anwendertechnischen Gründen Kompatibilität zu den 5¹/₄″-Laufwerken zu fordern ist, war ein ausreichendes Niveau der Speicherdichte, insbesondere eine Längsdichte um 500 Bit/mm, Voraussetzung dafür, daß sich diese Geräteklasse etablieren konnte. Aus diesem Grunde und unter Berücksichtigung künftiger Anforderungen sind überwiegend Metallschichtplatten in Gebrauch, die größtenteils in Sputtertechnologie hergestellt werden. Der 3,5″-Formfaktor mit den Abmessungen 41 mm × 102 mm × 146 mm ist wie üblich durch die Diskettenspeicher vorgegeben und gestattet, beim jetzigen Stand der Technologie bis 4 Platten unterzubringen. Mit einer Speicherdichte von 21 000 Bit/mm² bei k = 1 ergibt das als gegenwärtigen Spitzenwert 53 MByte. Während bislang in der 3,5″-Geräteklasse ausschließlich die Schrittmotorpositionierung mit mäßigen Zugriffszeiten zur Anwendung kam, hat nunmehr die Einführung des Tauchspulantriebs begonnen. Damit wurden bereits mittlere Positionierzeiten von 25 ms erreicht.
Die geringen Abmessungen der 3,5″-Laufwerke haben 1986 zu einer interessanten Lösung für die Leistungserhöhung des weitverbreiteten IBM-PC geführt. Auf eine Steckkarte der Größe 106 mm × 342 mm zusammen mit dem Controller montiert, läßt sich der Rechner auf einfachste Weise durch einen Plattenspeicher der Kapazität 10 MByte oder 20 MByte (formatiert) aufwerten /9/. Da die Leistungsaufnahme nur ca. 10 W beträgt, kann die Stromversorgung direkt über die Betriebsspannungen des Rechners erfolgen.
Es ist zu erwarten, daß in Kürze 3,5″-Laufwerke Kapazitäten bis 100 MByte erreichen. Damit werden in diesem Bereich im Verlaufe der nächsten Jahre die 5¹/₄″-Laufwerke weitgehend abgelöst. Die weiteren Entwicklungsmöglichkeiten zeichnen sich mit der erfolgten Ankündigung eines 170-MByte-Gerätes ab.

1.3.4. Effektivierung des Datenaustausches

Das gestiegene Leistungsvermögen der Mikrorechner und der Schaltkreistechnologien ermöglicht und erfordert, effektivere Methoden des Datenaustausches auch in der Welt der Mikrorechner-Peripherie anzuwenden. Dementsprechend haben in den letzten Jahren das SCSI (Small Computer System Interface) bzw. der SCSI-Peripheriebus nach ANSI ANSC X 3T9.2 starke Verbreitung gefunden /10/. An den SCSI-Bus sind 8 Geräte verschiedener Art anschließbar. Der Datenverkehr erfolgt über komplexe Kommandos mit einer Geschwindigkeit bis 1,5 MByte/s asynchron oder bis 4 MByte/s synchron. Die Ausnutzung dieser hohen Übertragungsraten erfordert bei den motorischen Speichergeräten die Zwischenschaltung von Halbleiter-Pufferspeichern. Ein weiterer Effektivitätsgewinn läßt sich durch überlappten Zugriff erreichen, sofern mehrere adressierbare Geräte aktiv sind.
Die Verwendung der SCSI-Schnittstelle bedeutet zugleich, daß ein Großteil der Controllerfunktionen in das Speichergerät verlagert wird. Das führt zu einer Entlastung des Hostrechners und zu einem Peripheriegerät mit „Intelligenz". Bei den Winchesterspeichern hat die SCSI-Schnittstelle vor allem für 5¹/₄″-Laufwerke Bedeutung erlangt, wird aber auch schon in die 3,5″-Geräteebene eingeführt. Durch den Einsatz hochintegrierter Spezialschaltkreise kann auch mit dieser Erweiterung der jeweilige Formfaktor eingehalten werden.
Eine Verkürzung der effektiven Zugriffszeit läßt sich durch das Prinzip des Cache-Speichers erreichen. Hochleistungs-Controller ermöglichen den Anschluß von 4 Laufwerken mit ESDI-Schnittstelle (Enhanced Small Disc Interface) und arbeiten mit Cache-Speichern

Tafel 1 Charakteristik der Geräteklassen bei Winchesterspeichern

Plattengröße	356 mm (14″)	356 mm (14″)	200 mm/210 mm (8″)	130 mm (5¹/₄″)	95 mm (3,5″)
Anwendung	EDVA	Kleinrechner	Supermikros	AC, PC	
Bauform	Standgerät	19″-Einschub	Einbaulaufwerk / Subsystem		
Kapazität [MB]	300–5000	160–800	50-800	25–760	12–170
Masse [kg]	300–500	40–60	6–14	1,5–5	1
Volumen [dm³]	800–1900	40–85	8–11	1,2; 2,4	0,6

von 1 MByte, aufrüstbar bis 16 MByte /10/, /11/.
Die Preis- und Kapazitätsentwicklung bei den 5¹/₄"- bis 8"-Laufwerken ermöglicht, unter Verwendung mehrerer Laufwerke Speichersubsysteme mit Kapazitäten > 1 GByte zu realisieren und mit Vorteil bei Kleinrechnern und mittleren EDVA anstelle von 14"-Geräten einzusetzen.

2. Back-up-Speicherung

2.1. Anwendungstechnische Erfordernisse

Mit dem breiten Einsatz immer größerer Festplattenspeicher wächst der Bedarf an adäquaten Mitteln zur Datensicherung, zum Datenaustausch und zur Datenarchivierung. Dementsprechend hat die back-up-Speichertechnik auf der Grundlage von Magnetbandkassetten stark an Bedeutung gewonnen /12/. Diese Geräte arbeiten aus Kosten- und Geschwindigkeitsgründen im streaming-Modus, das heißt, der Datenaustausch erfolgt file orientiert in größeren Einheiten.

2.2. Back-up-Speicherung bei Mikrorechnern

In der Regel kann nur bei Mikrorechnern und Personalcomputern mit Winchesterspeichern < 30 MByte auf einen speziellen back-up-Speicher neben der Diskette verzichtet werden. Das typische back-up-Medium für Personal- und Arbeitsplatzcomputer im gehobenen Leistungsbereich sowie für Supermikros ist gegenwärtig die ¹/₄"-Bandkassette. Die Aufzeichnung erfolgt seriell im Mehrspur-Serpentine-Verfahren auf γ-Fe_2O_3-Partikelbänder bei Bandgeschwindigkeiten um 2 m/s. Mit 4, 9 und 154 Spuren werden Speicherkapazitäten um 30, 60 und 125 MByte (formatiert) erreicht. Das Datenformat ist nach QIC-24 standardisiert. Die Systemintegration ist einfach, da üblicherweise der 5¹/₄"-Formfaktor der Disketten- und Winchesterlaufwerke einschließlich der slim-line-Ausführung eingehalten und neben dem Interface nach QIC-02 in steigendem Maße der SCSI-Anschluß ermöglicht wird. Um in Grenzfällen die Anschaffung eines speziellen back-up-Speichers zu vermeiden, wurden Kassettenplattenspeicher und hochdichte Disketten in spezieller Verpackung im 5¹/₄"- und 8"-Format entwickelt. Der Marktanteil dieser Erzeugnisse ist aber gering.

2.3. Hochleistungssysteme

Für EDVA und leistungsstarke Kleinrechner sind schnelle back-up-Speicher auf der Basis von ¹/₂"-Magnetbändern erforderlich. Mit dem neuen System 3480 von IBM wurde auch in diesem Bereich der Übergang zur Kassettentechnik eingeleitet. Die Kassette mit den Abmessungen 100 mm × 125 mm × 25 mm speichert 200 MByte auf einem ¹/₂"-CrO_2-Partikelband. Mittels der 18-Spur-Parallelaufzeichnung wird eine Datenübertragungsrate von 3 MByte erreicht. Eine weitere Erhöhung der Speicherdichte und -kapazität ist in den kommenden Jahren zu erwarten.
Verschiedene Aktivitäten sind darauf gerichtet, den Anwendungsbereich dieser Kassette durch die kostengünstigere serielle oder teilparallele Aufzeichnung zu verbreitern und Laufwerke im 5¹/₄"-Formfaktor zu entwikkeln.

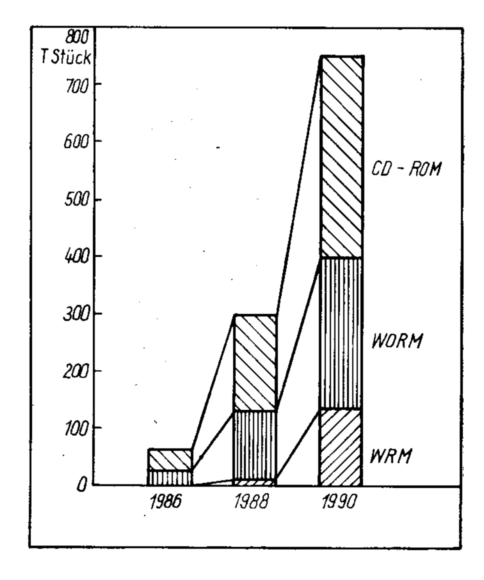


Bild 5 Entwicklung der Produktionsstückzahlen bei optischen Plattenspeichern

3. Digital-optische Plattenspeicher

3.1. Anwendungsaspekte

Nach der erfolgreichen Einführung der Compact-Disk im Jahre 1982 für die digitale Tonaufzeichnung höchster Qualität waren die nachfolgenden Aktivitäten der einschlägigen Industrie darauf angelegt, die optische Speicherung für die Daten- und Informationsverarbeitung nutzbar zu machen. Im Ergebnis dessen entstanden neue System- und Anwendungslösungen, ohne daß merkliche Rückwirkungen auf die Position der magnetomotorischen Speicher festzustellen sind. Die technische Entwicklung der digital-optischen Speicher vollzieht sich in 3 Kategorien:
– CD-ROM mit einer wechselbaren, unmittelbar aus der Compact Disk abgeleiteten Platte, die fest eingeprägte Daten trägt, welche nur gelesen werden können
– WORM (write once, read mostly) mit einer wechselbaren optischen Platte für einmalige Aufzeichnung durch den Anwender
– WRM (write/read memory) mit einer wechselbaren optischen Platte, die wie ein magnetisches Speichermedium Aufzeichnung und Wiedergabe in beliebiger Folge ermöglicht.

Die weitere Analyse muß sich auf einige anwendungstechnische Aspekte beschränken. Gegenwärtig sind international in den Kategorien CD-ROM und WORM sehr hohe Zuwachsraten zu verzeichnen, während WRM noch als Labor- und Entwicklungsmuster angesehen werden müssen. Prognosen besagen /13/, daß 1991 weltweit das Produktionsvolumen 1 Mio. Stück bei einem Marktwert von 2 Mrd. Dollar überschreitet (Bild 5). Trotz der erheblichen Steigerung dürfte diese Summe nur etwa 5 % des Marktwertes der magnetomotorischen Plattenspeicher ausmachen.

3.2. CD-ROM

CD-ROM bieten bei einem Plattendurchmesser von 120 mm eine Speicherkapazität von 200 ... 600 MByte pro Seite. Berücksichtigt man, daß die Plattenkassette für einen Preis von 60 bis 130 Dollar herstellbar ist und die Laufwerke im OEM-Vertrieb für 400 bis 900 Dollar gehandelt werden /14/, ergibt sich eine einzigartige Möglichkeit, sehr große Datenmengen dezentral auf Personalcomputern oder Arbeitsstationen verfügbar zu machen. Da die Laufwerke im 5¹/₄"-Formfaktor und mit SCSI-Schnittstelle zu haben sind sowie Standards für das Datenformat existieren, wird die Systemintegration sehr unterstützt. Es wird prognostiziert, daß 1990 5 % aller Personalcomputer mit Laufwerken für die optische Platte ausgestattet sein werden. Als Anwendungen kommen vor allem in Frage: Kataloge jeder Art, technische Dokumentationen, Serviceunterlagen, Nachschlagewerke, Lehrprogramme, Rechnerprogramme mit Dokumentationen usw. Die große Plattenkapazität läßt es zu, auch bildhafte Darstellungen abzuspeichern. Damit ist eine gewisse Alternative zur Informationsspeicherung auf Papier und Mikrofilm mit dem Vorteil des wahlfreien, rechnergestützten Zugriffs gegeben.

3.3. WORM

Die ersten für die digitale Informationsverarbeitung einsetzbaren optischen Speicher waren WORM-Platten mit einem Durchmesser von 12". Sie bieten eine wahlfrei zugreifbare Kapazität von 1 ... 4 GByte mit einer Positionierzeit > 100 ms. Die wesentlichste Zielstellung für die Anwendung waren zunächst Dokumentenspeichersysteme für die Büroautomaten, teilweise mit Kapazitäten > 100 GByte durch automatische Plattenwechselmechanismen. Außer den für die CD-ROM genannten Anwendungen kommen Einsatzmöglichkeiten in der Primärspeicherung von Objekten für die Bildverarbeitung, im Finanzwesen, bei Versicherungen, in der Medizin und generell in der Langzeit-Datenarchivierung hinzu. Der letzte Fall stellt im Prinzip eine Alternative zur Magnetbandspeicherung dar. Diese interessante Anwendung hat sich aber durch die fehlende Standardisierung und neue, leistungsfähige Kassettenbandgeräte (s. Pkt. 2.3) nicht im erwarteten Maße entwickelt. Gegenwärtig zeichnet sich ein Entwicklungsschwerpunkt bei Laufwerken und Plattenkassetten im 5¹/₄"-Formfaktor mit ESDI- oder SCSI-Schnittstelle ab, wobei Kapazitäten zwischen 300 und 800 MByte zur Diskussion stehen /14/, /15/.

3.4. WRM

Bei den löschbaren optischen Speichern stehen die 3,5"- und 5¹/₄"-Formate im Mittelpunkt des Interesses. Das magnetooptisch arbeitende Entwicklungsmuster von Verbatim speichert auf einer 3,5"-Platte 40 MByte. Diese Tendenz zu kleinen Formaten erklärt sich anwendungsseitig aus dem potentiellen Vermögen, im unteren Leistungsbereich der Personal- und Arbeitsplatzcomputer Festplatte, Diskette und back-up-Kassette durch ein einziges hochdicht speicherndes Medium ersetzen zu können, wobei in diesem Bereich die noch relativ großen Zugriffszeiten optischer Speicher am wenigsten stören. Einsatzreife Laufwerke wurden in diesem Jahr vorgestellt.

Ausblick

Das Leistungsvermögen und die Einsatzbreite der magnetomotorischen Speicher werden in den nächsten Jahren weiter zu-

nehmen. Bei Winchesterspeichern steigen die Speicherdichte und damit die Kapazitäten vor allem bei den 3,5"-, 5¼"- und 8"-Laufwerken weiter an. Es ist wahrscheinlich, daß mit der konventionellen Längsspeicherung Dichten von 10^5 Bit/mm^2 überschritten werden. Man kann davon ausgehen, daß sich bei den 5¼"- und 8"-Formaten die begonnene Entwicklung zu intelligenten Laufwerken, zum Einsatz leistungssteigernder elektronischer Mittel und zu Subsystemen mit mehreren Winchestermoduln fortsetzt. Die Position der Senkrechtspeicherung bei der Fortentwicklung der Plattenspeicher läßt sich gegenwärtig nur schwer bestimmen.

Unbestritten ist die wachsende Bedeutung der digital-optischen Speicher. Gravierende Rückwirkungen auf die magnetischen Medien sind vorerst nicht zu erwarten. Neben der technischen und technologischen Beherrschung der lösch- und umschreibbaren Platte müssen die Zugriffszeiten verringert und die Datenübertragungsraten erhöht werden.

Literatur

/1/ Laub, L.: The Evolution of Mass Storage. Byte 11 (1986) 5, S. 161–166, 168, 170–172

/2/ Salzmann, G.: Plattenspeicher – Stand und Entwicklungstendenzen. J. Inf. Rec. Mater. 14 (1986) 4, S. 245–255

/3/ Arnoldussen, Th. C.: Thin-Film Recording Media. Proc. IEEE 74 (1986) 11, S. 1526–1539

/4/ Salm, J.; Steinbeiß, Chr.: Kohlenstoffschichten als verschleißarme Schutzschichten für magnetomotorische Signalspeicher. Beitrag zur 7. Konferenz „Magnetische Signalspeicher", Neubrandenburg 1987

/5/ Salzmann, G.: Zur Testung der Oberflächenbeschaffenheit rotierender Scheiben im Submikrometerbereich. Beitrag zur 7. Konferenz „Magnetische Signalspeicher", Neubrandenburg 1987.

/6/ Waid, D. D.: Thin Film Challenges Oxide Media. Mini-Micro Systems. 18 (1985) 16, S. 95–98, 101–102

/7/ Fuchs, H.: Fertigung der Dünnfilmköpfe für Plattenspeicher. Feinwerktechn. u. Meßtechn. 94 (1986) 1. S. 47–48

/8/ Drive Includes ESDI or Embedded SCSI, packs 760 MBytes into a 5¼"-in. Size. EDN 31 (1986) 26, S. 62

/9/ Waller, L.: Hard Cards Roar along Despite Nagging Questions. Electronics 59 (1986) 5, S. 46–47

/10/ Wright, M.: Brigded SCSI Controllers Remain Viable Regardless of Emerging Embedded Controllers. EDN 32 (1987) 4, S. 69–76

/11/ ESDI-Controller mit Cache-Speicher. Elektronik 35 (1986) 17, S. 34–35

/12/ Antonuccio, A.: Tape Backup Systems. Byte 11 (1986) 5, S. 277–232

/13/ Optical-Disk-Drive Market to reach 2 Billion by 1991. EDN 31 (1986) 26, S. 179

/14/ Leibson, St. H.: Optical-Disk Drives Target Standard 5¼"-in. Sites. EDN 31 (1986) 26, S. 42–50

/15/ Zeltwanger, H.: Optische Speicher: Einmal schreiben, mehrmals lesen. Elektronik 35 (1986) 19, S. 34–35

/16/ Völz, H.: Stand der Anwendung optischer Speicher – insbesondere auf rotierenden Medien. Nachrichtentechnik Elektronik, Berlin 37 (1987) 5, S. 164–168

✉ KONTAKT ☏

VEB Robotron-Elektronik Dresden, Abt. E5E, PSF 330, Dresden 8012; Tel. 4872423

Optische Plattenspeicher[1]

František Marek, Ostrava (ČSSR)

1. Optische Speicherung – Wirkungsweise und Eigenschaften

Für die optische Speicherung /1/ werden als Speichermedium Plast-Disketten verwendet, die mit einer lichtempfindlichen Schicht versehen sind. Es werden Telluroxide, Gold- oder Silberlegierungen, Silberhalogenide u. a. verwendet.

Bei der Datenspeicherung auf ein optisches Medium werden mittels Laserstrahlen in die reflektierende Schicht ca. 1 μm große Löcher eingebrannt, so daß die Reflexion der Oberfläche geändert wird. Bei der Abtastung wird durch Auswertung der Signale in einem Detektor die Intensität des reflektierten Lichtes ermittelt; durch Unterscheiden von zwei Niveaus kann festgestellt werden, ob es sich um eine logische Null oder Eins handelt (Bild 1).

Der Vorteil der optischen Disketten ist z. Z. vor allem die hohe Aufzeichnungsdichte, insbesondere im Vergleich mit der magnetischen Aufzeichnung, wo mit Ausnahme der vertikalen Aufzeichnung nur kleine Reserven vorhanden sind. In Anbetracht der vorausgesetzten schnellen Entwicklung der Fertigungstechnologien von Speichermedien und Speicher-Laufwerken werden die optischen Speicher einen sehr niedrigen Aufwand für die Speicherung eines Bits aufweisen (Bild 2).

Die gegenwärtige Entwicklung ermöglicht bereits eine bedeutende Verringerung der Fehlerhaftigkeit der Speicherung, so daß das Auftreten fehlerhafter Bits bereits im Verhältnis 1 : 10^{12} angeführt wird; das ist ein Wert, der auch bei anspruchsvollen Anwendungen im Fall klassischer Speichermedien erreicht wird.

Ähnlich wie bei magnetischen Disketten bleibt das Problem der Nichtkompatibilität der optischen Speicher durch Anwendung von Speichern mit unterschiedlichen Durchmessern und Kapazitäten.

Vorteile gegenüber magnetischen Speichern sind unter anderem die Widerstandsfähigkeit gegen Staub, Rauch, Anfassen sowie eine lange Lebensdauer.

2. Optische Speicher mit nichtlöschbarer Aufzeichnung – OROM

Bei Anwendung optischer Disketten für nur eine Aufzeichnung stellt die hohe Kapazität einen bedeutenden Vorteil nicht nur auf dem Gebiet der Videoaufzeichnung oder Dokumentenarchivierung dar, sondern direkt in der Datenverarbeitung, insbesondere bei der Programmspeicherung und Abspeicherung großer Datenbasen. Deshalb erschien bereits 1984 auf dem Markt eine Reihe von Speichern für nichtlöschbare optische Aufzeichnung, bezeichnet mit der Abkürzung OROM (optical read only memory).

Die vorläufig größten optischen Disketten mit 356 mm Durchmesser (14" werden beim System der optischen Speicherung der Firma Storage Technology Corp. /2/ angewendet. Sie haben eine Kapazität von 4 GByte, die Übertragungsgeschwindigkeit beträgt 3 MBit/s. An das Steuerwerk können bis zu 8 Laufwerke angeschlossen werden.

Die bereits erwähnte japanische Firma Hitachi /3/ führte Ende 1984 das Laufwerk OC-301 vor, das eine 305 mm (12") große optische Diskette mit einer Kapazität bei beidseitiger Aufzeichnung von 2,6 GByte anwendet. Die Drehzahl der Diskette beträgt 600 U/min^{-1}, die Übertragungsgeschwindigkeit 440 KBit/s. Auf jeder Seite der Diskette befinden sich 40000 Speicherkurven mit einer durchschnittlichen Zugriffszeit von 200 ms.

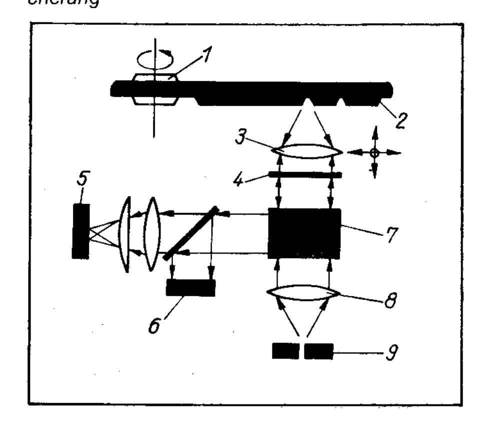
Thompson – CSF Communication brachte 1984 das Speicher-Laufwerk Gigadisc 1001 (Bild 3) auf den Markt, eine optische Diskette mit 305 mm Durchmesser und einer Kapazität von 2 GByte, Masse 1,3 kg (Bild 4). Die amerikanische Firma Reference Technology /4/ begann Ende 1984 Speicher-Laufwerke zu liefern mit der Bezeichnung Data Drive

---

[1] Der Artikel ist ein leicht gekürzter Nachdruck aus der tschechoslowakischen Zeitschrift „automatizace" 2/1986. Wir danken Autor und Redaktion für die Erlaubnis zur Veröffentlichung.
Die Übersetzung besorgte Paul Matuschek. Red.

Bild 1 Schematische Darstellung einer optischen Diskette
1 – Antriebsmechanismus, 2 – optisches Medium, 3 – Objektiv, 4 – Platte, 5 – Detektor für Scharfeinstellung, 6 – Detektor für Spureinstellung, 7 – Unterbrecher des polarisierten Strahles, 8 – Kolimator, 9 – Laserdiode

Bild 2 Vergleich der Technologien der Datenspeicherung



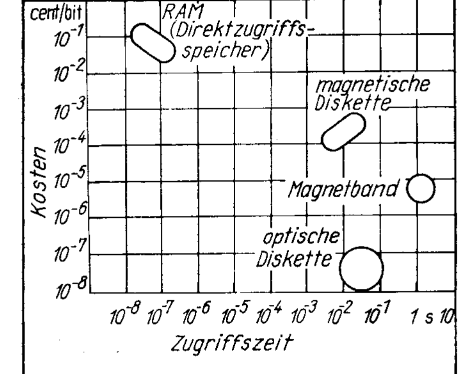




Bild 3 Laufwerk für optische Disketten als Tischgerät

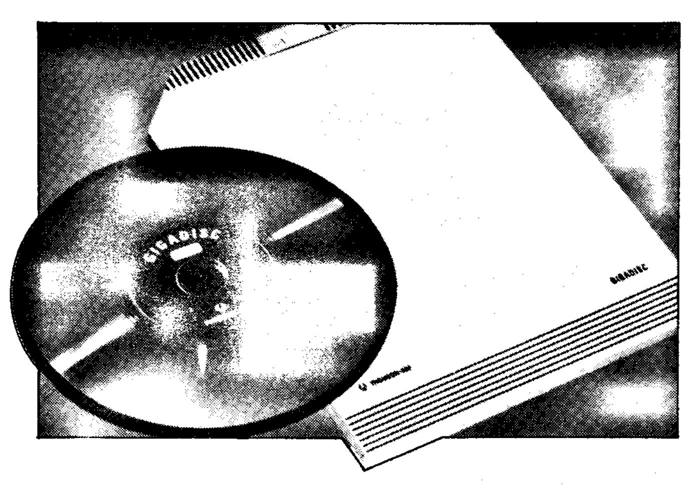


Bild 4 Optische Diskette von Thomson CSF

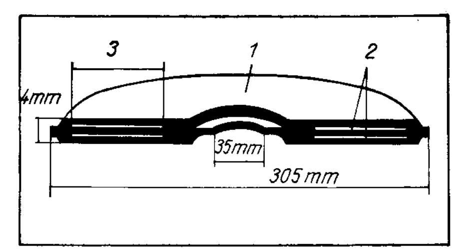


Bild 5 Schnitt einer optischen Diskette von 3M.
Das mit Rillen versehene Plastmaterial besitzt 1000 Å tiefe Rillen im Abstand von 1,67 μm. Jede Rille enthält vorgestanzte vorläufige Formatinformationen, Angaben zum Sektor und zur Spur.
1 – Laminat-Schutzschicht, 2 – Luftspalt, 3 – Speicherbereich

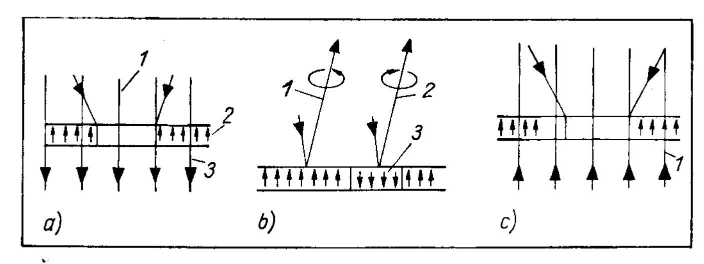


Bild 6 Magnetooptische Speichermethode
a) 1 – Laser, 2 – magnetisches Medium, 3 – Magnetfeld
b) 1 – normal reflektierter Strahl, 2 – optisch reflektierter Strahl, 3 – Domäne des umgekehrten Magnetismus
c) 1 – Magnetfeld

und den Abmessungen 214 × 406 × 600 mm^3 und einer Masse von 32 kg. Diese Speicher-Laufwerke verwenden ebenfalls optische Disketten mit einer Kapazität von 2 GByte, aber mit 51 000 Speicherspuren. Die Diskette hat eine Drehzahl von 1800 U/min^{-1}, die durchschnittliche Zugriffszeit beträgt 151 ms. Die Firma 3M begann mit der Lieferung von optischen Disketten mit 305 mm Durchmesser in zwei Varianten (Bild 5), beidseitiger Aufzeichnung und einer Kapazität von 1,2 GByte.
Laser Data Inc. kündigte die Herstellung optischer Disketten mit einer Kapazität von 800 MByte pro Diskette /5/ an, die für Personalcomputer geeignet sind.
Information Storage Inc. inseriert das Speicher-Laufwerk 525 WC /6/, welches eine optische Diskette mit 133 mm Durchmesser (5¼") und einer Kapazität von 100 MByte, einer Übertragungsgeschwindigkeit von 2,5 MBit/s und einer durchschnittlichen Zugriffszeit von 200 ms benutzt. In der Perspektive wird mit einem Preis von 600 Dollar für das Speicher-Laufwerk und 40 Dollar für die optische Diskette gerechnet.
Vollständigkeitshalber sei auch die Sortimentserweiterung bei Sony und Philips /5/ erwähnt, die für die Computertechnik Compakt-Disks mit 12 cm Durchmesser (4,72"), einer Kapazität von 550 MByte, durchschnittlicher Zugriffszeit von 0,5 s und einer Übertragungsgeschwindigkeit von 1,44 MBit/s liefern.

3. Optische Disketten mit löschbarer Aufzeichnung

Die Verbindung einer hohen Speicherdichte mit entsprechendem Anwachsen der Speicherkapazität und die Möglichkeit einer mehrmaligen Aufzeichnung würde verständlicherweise die Anwendung der optischen Aufzeichnung in der Computertechnik deutlich verbessern. Zu diesem Zweck beginnt man, in der Praxis zwei weiter beschriebene Methoden zu forcieren /7/, /8/.

3.1. Magnetooptische Methode

Bei dieser Methode wird die Temperaturabhängigkeit der magnetischen Koerzitivkraft beim Lesen, Löschen und Neuaufzeichnen mittels einer Laserdiode genutzt. In der Aufzeichnungsphase wird in das Medium ein Magnetfeld mit einigen hundert Oersted geleitet. Der Lichtstrom aus der Laserdiode bildet in der Speicherschicht kleine Öffnungen in der Größe von 1 μm, wobei gleichzeitig die Aufzeichnungsstelle auf eine Temperatur erwärmt wird, bei der die Koerzitivkraft gleich Null ist, so daß das Medium am leichtesten magnetischen Änderungen zugänglich ist. Durch das Einwirken des Magnetfeldes kommt es zu einer Aufzeichnung des entsprechenden Bits, ähnlich wie bei der magnetischen Aufzeichnung.
Zum Lesen der Aufzeichnung wird der sogenannte Kerr- oder Faraday-Effekt genutzt, der auf dem Drehen der Polarisationsebene des Lichtes durch ein Magnetfeld beruht. Polarisiertes Licht dreht sich im oder gegen den Uhrzeigersinn, je nach Richtung des Magnetismus. Mit Hilfe eines Analysators wird der Grad der Polarisation ermittelt und dadurch die Aufzeichnung in Form einer Null oder Eins ausgewertet.
Das Löschen der Aufzeichnung kann auf zweierlei Art ausgeführt werden. Einerseits geschieht dies lokal, das heißt durch Erwärmung und Einwirkung eines Magnetfeldes, das eine Aufzeichnung bewirkt (Bild 6). Andererseits mittels massenhaften Löschens aller Aufzeichnungen auf der Diskette.
Bei der Größe einer Aufzeichnungsöffnung von 1 μm kann durch diese Methode eine

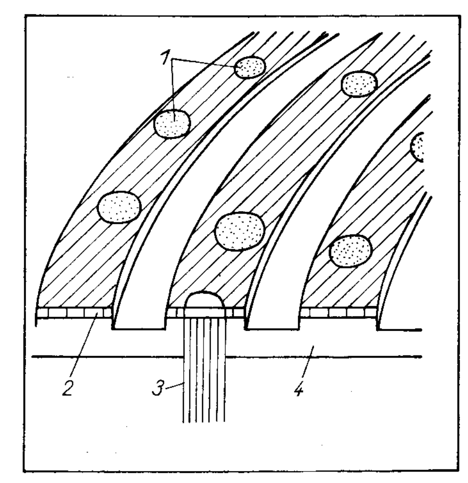


Bild 7 Speicherung mittels Methode der Änderung der Kristallphase einer Speicherschicht in eine amorphe Phase
1 – gespeicherte Bits, amorphe Phase, 2 – Speicherschicht, Kristallphase, 3 – Laserstrahl, 4 – Grundmaterial der Diskette

Speicherdichte von $5 \cdot 10^8$ Bit/ cm^2 erreicht werden. Die erste praktische Anwendung dieser Methode stellen optische Disketten mit einer Speicherschicht auf Terbium-, Eisen- und Kobaltbasis dar, die gemeinsam von den Firmen Sony und Kokusai Denshin Denwa hergestellt werden. Diese Disketten können für eine digitale sowie analoge Aufzeichnung angewendet werden.

3.2. Methode der Änderung der Kristallphase einer Speicherschicht in eine amorphe Phase

Diese Methode wurde von der Firma Matsushita realisiert, wobei als empfindliche Speicherschicht eine Tellurlegierung mit kleinen Mengen Germanium, Indium und Blei angewendet wird. Bei einer Aufzeichnung mittels Laserstrahl (Bild 7) ändert sich die Kristallphase mit großer Reflexion in eine amorphe mit niedriger Reflexion. Die Aufzeichnung und das Lesen werden durch einen Laser mit 830 nm Wellenlänge und 8 mW Leistung für die Aufzeichnung und 1 mW beim Lesen realisiert. Das Löschen geschieht durch einen zweiten Laser mit 780 nm Wellenlänge und 10 mW Leistung.
Eine praktische Realisierung stellt das Laufwerk von Panasonic OMDR (optical memory disk recorder) dar, welches eine 8"-Diskette mit einer Kapazität von 700 MByte/Seite und einer Drehzahl von 1800 U/min^{-1} anwendet. Der Vorteil dieser Geräte ist eine einfachere Konstruktion als im Fall der magnetooptischen Aufzeichnung. Perspektivisch wird die Anwendung eines Mediums auf der Basis eines Polymerfarbstoffes erwogen, dadurch würden Probleme beseitigt werden, die mit der Verarbeitung von Rohstoffen mit Giftwirkung (Tellur) zusammenhängen.

4. Ausblick

Optische Disketten finden bereits breite Anwendung auf dem Gebiet der Konsumgüterelektronik (Videoaufzeichnung) und bei der Archivierung von Dokumenten. Dank der außerordentlichen Speicherdichte, die eine Kapazitätserhöhung bei gleichzeitiger Verringerung der Abmessungen und der Masse ermöglicht, behaupten sie sich immer mehr auch in der Rechentechnik, vor allem bei der Speicherung von Programmen und beim Aufbau umfangreicher Datenbanken.
Ihre breite Anwendung ist allerdings von der Lösung des Problems des Löschens der alten Aufzeichnung und der Realisierung einer neuen abhängig.

Literatur

/1/ Byte (1983) 3, S. 86–106
/2/ Byte (1984) 10, S. 221
/3/ Byte (1985) 2, S. 367
/4/ Byte (1985) 1, S. 418
/5/ Byte (1985) 1, S. 421
/6/ Computer Product News (1985) 5, S. 35
/7/ Byte (1984) 10, S. 215
/8/ Sdelovaci technika, Praha (1984) 11, S. 427

Einchipmikrorechner

Das Sortiment des VEB Mikroelektronik „Karl Marx" Erfurt

Volkmar Heilbock
VEB Mikroelektronik „Karl Marx" Erfurt

Vorbemerkungen

Der VEB Mikroelektronik „Karl Marx" Erfurt produziert seit 1984 ein umfangreiches Sortiment von Einchip-Mikrorechner-Schaltkreisen (EMR).
Basierend auf den beiden Basistypen dieses 8-Bit-Mikroprozessorsystems, dem U 881 D und dem U 882 M /1/, umfaßt die Einchip-Mikrorechner-Familie heute 7 Typvarianten in insgesamt 14 Bondversionen mit jeweils 3 möglichen Taktfrequenz-Obergrenzen. All diese Varianten sind hinsichtlich ihrer internen Struktur, ihrer statischen und dynamischen Kennwerte sowie ihrer Programmierung grundsätzlich identisch. (Siehe Bilder 1 bis 3 und Tafel 1.)

Die Einchip-Mikrorechner lassen sich in zwei Gruppen einteilen:

- *EMR mit internem ROM*

Diese Gruppe repräsentiert den **Einchip-Rechner im eigentlichen Sinne.**
Der Programmspeicher, der On-Chip-ROM, befindet sich hier in Form einer kundenspezifischen Maske (Bitmuster) auf dem Chip.
Die Herstellung dieser Maske ist mit relativ hohen Kosten verbunden, die der Anwender dem EMR-Hersteller einmalig zu erstatten hat.
Somit ergeben sich als sinnvolle Einsatzfälle fast ausschließlich Geräte mit relativ hohen Produktionsstückzahlen, wie z. B. Konsumgüter.

- *EMR mit externem ROM bzw. mit externem Programmstart*

Zu dieser Gruppe zählen die Einchip-Mikrorechner-„Entwicklungsversionen", deren Programmspeicher logisch *intern*, aber phy-

Tafel 1 Flags des EMR

Bedingungscode	Hex (oberes Nibbel)	Bedeutung	gesetzte Flags
(AT)	8	immer wahr	--
C	7	"Carry"	C = 1
NC	F	kein "Carry"	C = 0
Z	6	Null	Z = 1
NZ	E	nicht Null	Z = 0
PL	D	plus	S = 0
MI	5	minus	S = 1
OV	4	"Overflow"	V = 0
NOV	C	kein "Overflow"	V = 1
EQ	6	gleich	Z = 1
NE	E	nicht gleich	Z = 0
GE	9	größer als oder gleich	(S XOR V) = 0
LT	1	kleiner als	(S XOR V) = 1
GT	A	größer als	Z OR (S XOR V) = 0
LE	2	kleiner als oder gleich	Z OR (S XOR V) = 1
UGE	F	vorzeichenfrei größer als oder gleich	C = 0
ULT	7	vorzeichenfrei kleiner als	C = 1
UGT	B	vorzeichenfrei größer als	(C = 0 AND Z = 0) = 1
ULE	3	vorzeichenfrei kleiner als oder gleich	(C OR Z) = 1
NT	0	niemals wahr	--

Tafel 2 EMR-Sortiment des VEB Mikroelektronik Erfurt

	EMR mit externem ROM bzw. mit externem Programmstart					EMR mit internem ROM	
	ohne ROM		Interpreter für Tiny-MP BASIC			Kundenspezifisches Programm	
	Programmstart extern			Entwicklungsversion			
	bei erhöhtem Pegel an RESET-Eingang		nach normalem RESET	über Speicherport zugängliche ROM-Kapazität		On-Chip-ROM-Kapazität	
	auf Adresse %1012		auf Adresse %812	4 KByte	2 KByte	4 KByte	2 KByte
Normal-version Quarz direkt anschließbar	U 8611 DC 08/1	UB 8860 D	UB 8830 D	UB 8840 M	UB 8820 M	U 8611 DC08	UB 8810 D
Power-Down-Version Takt-generator erforderlich	UL 8611 DC 08/1	UB 8861 D	UB 8831 D	UB 8841 M	UB 8821 M	UL 8611 DC08	UB 8811 D

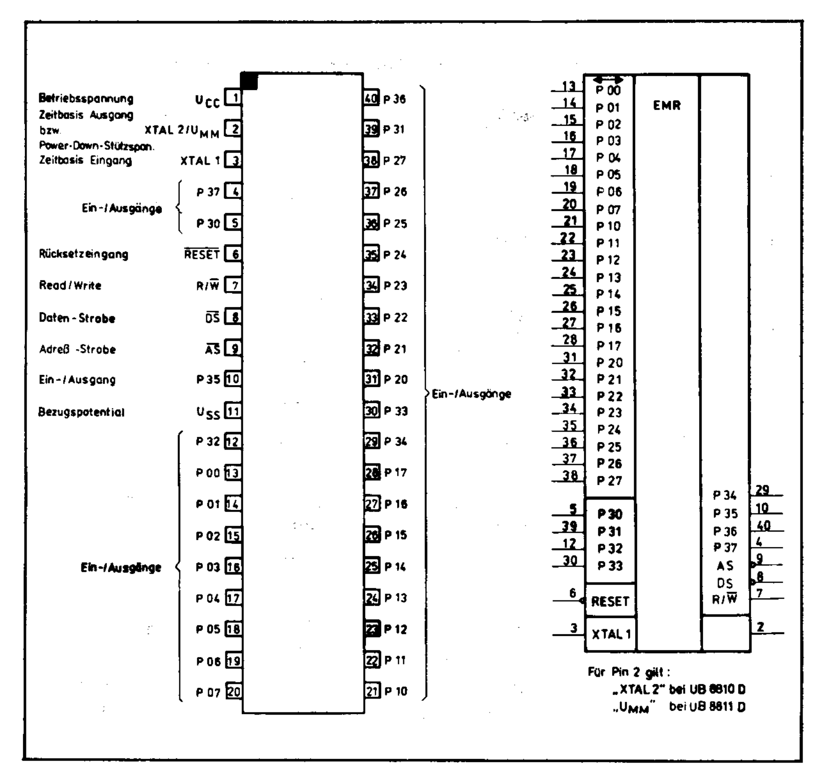


Bild 1 Pinbelegung und Schaltungskurzzeichen des UB 8810 D/UB 8811 D

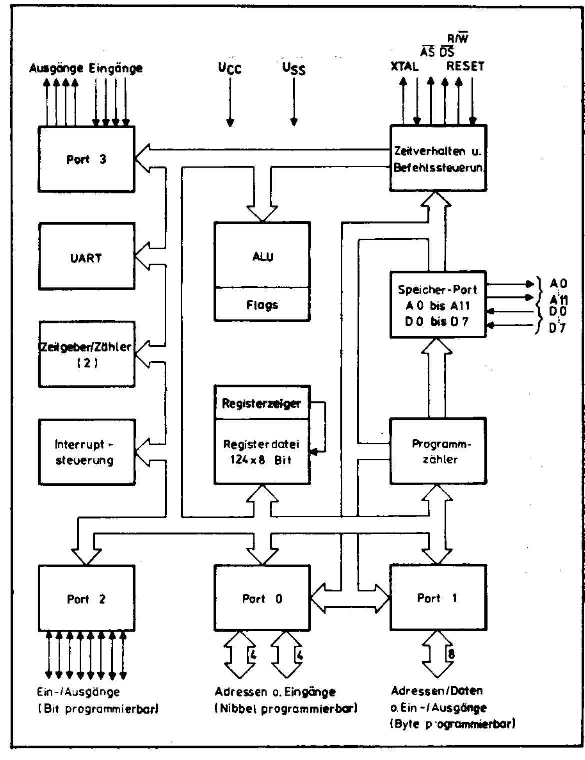


Bild 3 Blockschaltbild des EMR am Beispiel des UB 8820 M

sich *extern* angeordnet ist. Auch lassen sich dieser Gruppe all jene EMR mit On-Chip-ROM zuordnen, deren Bitmuster für eine universelle Verwendung vorgesehen ist und/oder deren Programmstart extern erfolgen kann.

Das Anwenderprogramm befindet sich hier außerhalb des Einchip-Mikrorechners und ist für diesen über das spezielle Speicherport bzw. über die als Adreß-/Daten-Bus konfigurierbaren Ports 0 und 1 zugänglich.

Sinnvolle Einsatzgebiete sind Geräte mit kleineren Produktionsstückzahlen, mit oft bzw. kurzfristig einer Änderung zu unterwerfenden Programmen sowie Lern-, Entwicklungs- und Emulationsmodule.

Tafel 2 gibt eine Übersicht zum derzeitigen Produktionssortiment des VEB Mikroelektronik „Karl Marx" Erfurt.

Die verwendeten Typbezeichnungen beziehen sich auf EMR mit einer maximalen Taktfrequenz von 8,0 MHz.

EMR mit internem ROM

- *UB 8810 D, UB 8811 D, U 8611 DC08, UL 8611 DC08*

Das auf dem Chip befindliche kundenspezifische Programm hat eine Kapazität von 2 KByte bzw. 4 KByte.

Eine externe Speichererweiterung kann mittels Nutzung von Port 0 und Port 1 als Adreß- und Datenbus vorgenommen werden. Dadurch ist die Schaffung eines zusätzlichen, direkt adressierbaren Speicherbereiches von 124 KByte (UB 8810 D, UB 8811 D) bzw. 120 KByte (U 8611 DC08, UL 8611 DC08) möglich.

Die Kennzeichnung des jeweiligen kundenspezifischen Bitmusters erfolgt durch eine 3stellige Zahl als Zusatz zur Typbezeichnung, z. B. UB 8810 D-003, UL 8611 DC08-002.

EMR mit externem ROM bzw. mit externem Programmstart

- *UB 8820 M, UB 8821 M, UB 8840 M, UB 8841 M*

Das logisch interne Anwenderprogramm wird physisch außerhalb des EMR angeordnet.

Bild 2 Pinbelegung und Schaltungskurzzeichen des UB 8820 M/UB 8821 M

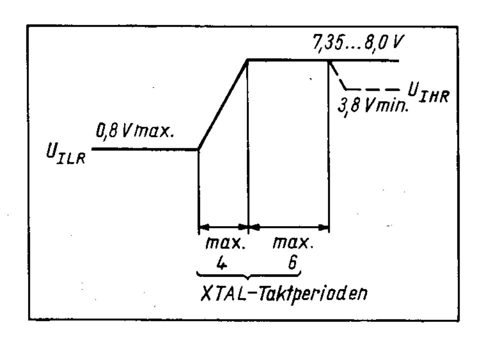


Bild 4 RESET-Pin-Beschaltung zum Eintritt in die Test-ROM-Initialisierungsroutine

Zu diesem Zweck verfügt der Einchip-Mikrorechner über 12 Adreß-Ausgänge und 8 Daten-Eingänge (Speicherports).
Eine bidirektionale Nutzung dieses Speicherports ist nicht möglich. Es wird ein spezielles Signal /MDS (Memory Data Select) bereitgestellt, welches zur Aktivierung eines angeschlossenen ROMs bzw. EPROMs (z. B. U 2716 C, U 2732 C) dient. Zu beachten ist jedoch, daß beim Lesen von *Interruptvektoren* im logisch internen Bereich das Signal /MDS nicht aktiviert wird! In diesem Fall sind der Chip-Select- und der Output-Enable-Eingang des EPROMs ständig mit LOW-Pegel zu beschalten.

- *UB 8830 D, UB 8831 D*

Diese EMR-Typen enthalten einen Interpreter für Tiny-MPBASIC /2/, /3/. Das interne Programm besteht aus einer Initialisierungsroutine und dem eigentlichen Interpreter, der mit Hilfe eines externen Programms gesondert aufzurufen ist. Die Initialisierungsroutine dient der Ausführung verschiedener Tests /4/, bevor ein Anwenderprogrammstart auf der externen Speicheradresse

%0812 (wenn Programmspeicherplatz %0812 RAM, also beschreibbar ist)
oder
%E000 (wenn Programmspeicherplatz %0812 ROM, also nicht beschreibbar ist)

ausgelöst wird. Außerdem wird der EMR auf *extended memory timing* programmiert.

Die Initialisierungsroutine hat folgendes Aussehen:

Adresse	Opcode	Befehl
000C	31 00	SRP #0
000E	3C 0F	LD R3,# %0F
0010	FF	NOP
0011	76 E3 04	TM R3, #4
0014	14 3C FF	LD R3, #%FF
0016	EB 05	JR NZ, #%5
0018	76 E3 04	TM R3, #4
001B	EB 20	JR NZ, Test
001D	E6, F8 B6	LD P01M, #%(2) 10110110
0020	E6 F7 08	LD P3M,#%(2) 00001000
0023	4C08	LD R4, #8
0025	5C 12	LD R5,#%12
0027	C2 64	LDC R6,§RR4
0029	60 E6	COM R6
002B	D2 64	LDC §RR4,R6
002D	C2 74	LDC R7,§RR4
002F	60 E6	COM R6
0031	D2 64	LDC §RR4, R6
0033	B2 67	XOR R6,R7
0035	31 F0	SRP #%F0
0037	ED E0 00	JP NZ,%E000
003A	8D 08 12	JP %812

Der Programmteil „Test" und seine Handhabung wird in /4/ ausführlich beschrieben.
Der Interpreter für Tiny-MPBASIC wird folgendermaßen aufgerufen:
① Laden der Startadresse des BASIC-Programms in die EMR-Register 6 und 7.
② Laden der Startadresse der Prozedurtabelle in die EMR-Register 8 und 9. Ist keine Prozedurtabelle vorhanden, dann sind die Register 8 und 9 zu löschen!
③ Aufruf des Interpreters mit den Befehlen
SRP #%10
CALL %7FD.
Somit stellt auch dieser Typ eine universell einsetzbare Variante eines Einchip-Mikrorechners dar.

- *UB 8860 D, UB 8861 D, U 8611 DC08/1, UL 8611 DC08/1*

Diese Typen werden auch als *ROM-los* bezeichnet.
Zusätzlich zum eigentlichen kundenspezifischen ROM ist in jedem Einchip-Mikrorechner mit *internem* Programmspeicher ein *Test-ROM* integriert. Dieses benötigt der Hersteller zu meßtechnischen Zwecken, um unter Umgehung des Kunden-ROMs sein *externes* Testprogramm starten zu können. bei den ROM-losen Typen wird jedoch ausschließlich der Test-ROM genutzt; Inhalt und Funktion des Kunden-ROMs bleiben dem Anwender weitgehend verborgen.
Hervorgerufen wird der externe Programmstart durch Beschalten des RESET-Pins mit einem Pegel von + 7,35 ... + 8,0 V nach Bild 4.
Beim UB 8860 D und UB 8861 D erfolgt der externe Programmstart auf der Adresse %812, beim U 8611 DC08/1 und UL 8611 DC08/1 auf der Adresse %1012.
Die Initialisierungsroutine im Test-ROM hat folgendes Aussehen:

Adresse	Opcode	Befehl
000C	E6 00 00	LD P0 #%00
000F	E6 F8 96	LD P01M #%96
0012	8D 08 12	JP Startadresse (UB 8860 D, UB 8861 D)
0012	8D 10 12	JP Startadresse (U 8611 DC08/1, UL 8611 DC08/1).

Literatur

/1/ Bankel, M.: Einchip-Mikrorechner U 881, U 882 und U 883. Radio, Ferns. Elektron., Berlin 34 (1985) 2, S. 81–84
/2/ Müller, S.: Einchip-Mikrorechner U 883 D interpretiert Tiny-MP-Basic. Radio Ferns. Elektron., Berlin 34 (1985) 3, S. 143–144
/3/ Müller, S.: Programmieren mit BASIC. REIHE AUTOMATISIERUNGSTECHNIK, Band 216, 1. Auflage. VEB Verlag Technik, Berlin 1985
/4/ Technische Beschreibung Einchipmikrorechner U 883. veb mikroelektronik „karl marx" erfurt, Erfurt 1985

⊠ KONTAKT ☏

VEB Mikroelektronik „Karl Marx" Erfurt, Abt. Applikation Bauelemente, Rudolfstraße 47, Erfurt, 5010; Tel. 51076 App. 25

CMOS-Technologien gewinnen weiter an Boden

Nachdem sich in den vergangenen Jahren CMOS-Technologien für den Anwendungsbereich der Höchstintegration durchgesetzt haben und heute die Herstellung solcher Spitzenprodukte der Mikroelektronik wie 4-Megabit-DRAM und 32-Bit-Einchiprechner ermöglichen, sind sie nun im Begriff, auch die letzte Domäne bipolarer Technologien ins Wanken zu bringen: den Bereich der Höchstgeschwindigkeitsbauelemente. Über geradezu sensationelle Fortschritte beim Vorstoß in den Pikosekundenbereich wurde auf der kürzlich in Bologna, Italien abgehaltenen 17th European Solid State Device Research Conference (ESSDERC) berichtet. Während bisher der Subnanosekundenbereich für die Gatterverzögerungszeit logischer Schaltungen im allgemeinen der schnellen ECL-Technik vorbehalten war, wurde nun gleich durch mehrere Entwicklungslabors der ECL-Geschwindigkeitsbereich mit CMOS-Gattern nicht nur erreicht, sondern teilweise sogar übertroffen. Lewis und Mitarbeiter vom Xerox Palo Alto Research Centre, USA, berichteten über eine Submikrometer CMOS-Technologie mit weniger als 100 ps Gatterverzögerungszeit und einen um mehr als 1 Größenordnung geringerem Verzögerungszeit-Leistungsprodukt (PDP) als vergleichbare ECL-Technologien. Hanafi und Mitarbeiter von IBM erreichten mit einer 0,5-µm-CMOS-Technologie Gatterverzögerungszeiten von 90 ps bei einem PDP von 17 fJ. Den absoluten Rekord für Siliziumbauelemente vermeldeten Xu Xiao-Li und Mitarbeiter vom Nanjing Institut of Technology, VR China, denen mit einer neuen Variante einer SOI-CMOS-Technologie Gatterverzögerungszeiten von 5,5 ps bei einem PDP von rund 4 fJ gelangen. Die minimalen Strukturabmessungen betrugen ebenfalls 0,5 µm. Noch geringere Abmessungen realisiert Sai-Halasz, IBM, der mit 0,07 µm Kanallänge die weltweit bisher kleinsten MOS-Transistoren zunächst noch nur in N-Kanaltechnik realisierte. Die minimale Gatterverzögerungszeit betrug 20 ps.
Damit dürfte die CMOS-Technologie in den nächsten Jahren auch für den Höchstgeschwindigkeitsbereich (VHSIC) kommerziell interessant werden.

Prof. Dr. Bernd Junghans

Programmieren mit MACRO-SM

Teil III

Dr. Thomas Horn
Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden

In Anweisungen mit den Bedingungen **DF** und **NDF** sind logische Ausdrücke mit den Operatoren **!** und **&** unter Verwendung von spitzen Klammern ⟨ ⟩ zulässig. Die Operatoren haben die gleiche Priorität und werden von links nach rechts abgearbeitet, falls durch eine Klammerung keine andere Abarbeitungsfolge vorgeschrieben wird.

Beispiel:
```
.IF DF M0! <GAMMA&DELTA>
...
...
.ENDC
```

Die Anweisungen des bedingten Blockes werden übersetzt, wenn das Symbol **M0** oder die Symbole **GAMMA** und **DELTA** definiert sind.
Die maximale Schachtelungstiefe der bedingten Blöcke beträgt 16. Beim Auftreten einer **.ENDC**-Anweisung außerhalb eines bedingten Blockes bzw. beim Überschreiten der maximalen Schachtelungstiefe wird der Fehlercode 0 angezeigt.

3.2. Die Anweisungen .IFF, .IFT und .IFTF

Die Anweisungen **.IFF**, **.IFT** und **.IFTF** unterteilen einen bedingten Anweisungsblock in Subblöcke und beziehen sich bei geschachtelten bedingten Anweisungsblöcken immer auf die getastete Bedingung der vorangegangenen **.IF**-Anweisung des gleichen Niveaus.
Die **.IFF**-Anweisung bewirkt das Übersetzen der nachfolgenden Anweisungen, wenn die getestete Bedingung den Wahrheitswert "false" ergab.
Die **.IFT**-Anweisung bewirkt das Übersetzen der nachfolgenden Anweisungen, wenn die getestete Bedingung den Wahrheitswert "true" ergab.
Die **.IFTF**-Anweisung bewirkt das Übersetzen der nachfolgenden Anweisungen unabhängig von der getasteten Bedingung. Die einzelnen Subblöcke werden jeweils durch eine nachfolgende **.IFF-**, **.IFT-**, **IFTF-** oder **.ENDC**-Anweisung gleichen Niveaus abgeschlossen.

Beispiel
```
SYMBOL=25
...
.IF    EQ SYMBOL-5
       ;TEST OB SYMBOL-5 GLEICH 0
       IST?
...    ;ANWEISUNGEN WERDEN
       NICHT UEBERSETZT
...
.IFF
...    ;ANWEISUNGEN WERDEN
       UEBERSETZT
...
.IFTF
...    ;ANWEISUNGEN WERDEN
       IMMER UEBERSETZT
...
.IFT
...    ;ANWEISUNGEN WERDEN
       NICHT UEBERSETZT
...
.ENDC
```

3.3. Die .IIF-Anweisung

Die **.IIF**-Anweisung gestattet die bedingte Übersetzung einer einzelnen Anweisung *statement*. Sie hat folgenden Aufbau:

.IIF *cond, arg[,arg], statement*

wobei die Bedingungen, Trennzeichen und Argumente denen der **.IF**-Anweisung entsprechen.
Bei den Bedingungen **B** bzw. **NB** sollte das Argument in spitze Klammern gesetzt werden, da sonst im Falle eines leeren Argumentes (blank) der Parameter statement als *arg* interpretiert wird.

Beispiel:
.IIF DF, ⟨FELD⟩ M0: MOV FELD,R0
Wenn das Symbol **FELD** definiert ist, wird vom Assembler die Anweisung
M0: MOV #FELD,R0
übersetzt. Andernfalls wird die Anweisung nicht übersetzt.

4. Die Makrotechnik

Bei der Programmierung in der Assemblersprache kann der Programmierer oft benötigte Anweisungsfolgen unter Anwendung der Makrotechnik als Makro definieren. Die Makrodefinitionen können Bestandteile des Quellprogramms sein oder in einer Nutzer- oder Systemmakrobibliothek zusammengefaßt werden. Die Bezugnahme auf die Makrodefinitionen erfolgt an den gewünschten Stellen im Programm durch sogenannte Makrorufe, die eine Generierung der in der Makrodefinition enthaltenen Quellanweisungen bewirken. Zur Anpassung des Makros an den jeweiligen Anwendungsfall besteht die Möglichkeit, daß in der Makrodefinition formale Argumente festgelegt werden, die bei der Makrogenerierung durch aktuelle Parameter ersetzt werden können. Die generierten Quellanweisungen werden als Makroauflösung bezeichnet.

4.1. Die Makrodefinition

Eine Makrodefinition beginnt mit einer Makroanfangsanweisung (**.MACRO**), die gleichzeitig die Musteranweisung für den Makroruf ist. Anschließend folgen die Modellanweisungen für die zu generierenden Quellanweisungen. Den Abschluß der Makrodefinition bildet die Makroendanweisung (**.ENDM**). Als Modellanweisungen können in den Makrodefinitionen alle Maschinenbefehle, Assembleranweisungen, Anweisungen der bedingten Übersetzung, Makrorufe und spezielle Anweisungen der Makrotechnik (z. B. **.PRINT**, **.ERROR**, **.MEXIT**, **.NTYPE**, **.NCHR**, **.NARG**) Anwendung finden. Eine Makrodefinition kann im Quellprogramm an beliebiger Stelle stehen, aber es ist zu beachten, daß sie vor ihrem ersten Aufruf bekannt sein muß.

4.1.1. Die .MACRO-Anweisung

Die Makroanfangsanweisung hat folgenden Aufbau:

.MACRO *symbol [,arg [,arg [,. . .]]]*
wobei
symbol – den Namen der Makrodefinition bzw. des Makrorufes festlegt,
, – ein zulässiges Trennzeichen (Komma, Tabulator und/oder Leerzeichen)
und
arg – ein folgendes Argument bedeuten.

Eine **.MACRO**-Anweisung kann beliebig viele formale Argumente enthalten, die durch wenigstens ein zulässiges Trennzeichen getrennt sein müssen, das heißt, es dürfen ein oder mehrere Tabulatoren oder Leerzeichen, aber nur ein Komma verwendet werden. Als formale Argumente werden Symbole verwendet, auf die sich die Modellanweisungen innerhalb einer Makrodefinition beliebig beziehen können, da sie im Namens-, Operations- oder Operandenfeld oder in Teilen von ihnen oder aber auch als komplette Quellanweisung benutzt werden dürfen. Diese Symbole sind nur interne Symbole einer Makrodefinition.

Beispiel:
.MACRO PUT EABER, LAENGE
Diese Anweisung leitet die Definition des Makros PUT mit den formalen Argumenten **EABER** und **LAENGE** ein.

4.1.2. Die .ENDM-Anweisung

Die Makroendanweisung hat folgenden Aufbau:
.ENDM *[symbol]*
wobei:
symbol – den Namen der zu beendenden Makrodefinition festlegt. Das Symbol hat den Charakter eines Kommentars.

Beispiel:
```
.MACRO   RETT1 P1
MOV      #P1,R5
.MACRO   RETT2
```

```
MOV     R0,(R5)+
MOV     R1,(R5)+
MOV     R2,(R5)+
.ENDM   RETT2 ;ENDE RETT2
MOV     R3,(R5)+
MOV     R4,(R5)+
.ENDM   RETT1 ;ENDE RETT1
```

Bei ineinander geschachtelten Makrodefinitionen kann die innere Makrodefinition erst dann aufgerufen werden, wenn die äußere Makrodefinition wenigstens einmal generiert wurde.

4.1.3. Die .MEXIT-Anweisung

Die **.MEXIT**-Anweisung beendet die Makrogenerierung, bevor das Ende der Makrodefinition erreicht wurde. Besonders vorteilhaft ist die Anwendung der **.MEXIT**-Anweisung in Verbindung mit der bedingten Assemblierung zum Zwecke der bedingten Makrogenerierung bzw. zum Abbruch der Makrogenerierung bei Fehlerbedingungen.

Beispiel:

```
.MACRO  BEISP P1,P2
...
.IF     B,P2
...
...
.MEXIT  ;ENDE MAKROGENERIE-
        ;RUNG, WENN P2 NICHT
        ;SPEZIFIZIERT WURDE
.ENDC
...
.ENDM
```

Wenn das formale Argument **P2** im Makroruf durch keinen aktuellen Parameter ersetzt wird, so werden die Anweisungen nach der **.IF**-Anweisung generiert und danach wird die Generierung abgebrochen. Andernfalls wird der bedingte Block übergangen.

4.1.4. Die .PRINT- und .ERROR-Anweisungen

Die **.PRINT**- und **.ERROR**-Anweisungen werden benutzt, um Mitteilungen in das Assemblerprotokoll auszugeben. Die Anweisungen haben folgendes Format:

.PRINT *[expr] [,text]*

.ERROR *[expr] [,text]*

wobei
expr – zulässiger Ausdruck und
text – die auszugebende Mitteilung bedeuten. Der Wert des Ausdrucks wird berechnet und in der Spalte des Objektkodes ausgedruckt. Dieser Wert kann als eine Mitteilungsnummer betrachtet werden. Durch den Text wird die Mitteilung in der Regel kommentiert. Der Unterschied zwischen den beiden Anweisungen besteht darin, daß bei der **.ERROR**-Anweisung zusätzlich noch das Fehlerkennzeichen P gedruckt wird.

4.2. Der Makroruf

Die Bezugnahme auf eine Makrodefinition erfolgt durch einen Makroruf, der eine Quellzeile mit folgendem Aufbau darstellt:
[symbol:] name liste aktueller parameter

Die Quellzeile kann durch ein Symbol im Namensfeld benannt sein. Im Operationsfeld steht ein Symbol, das den Namen der aufzurufenden und zu generierenden Makrodefinition (siehe **.MACRO**-Anweisung) angibt. Die Liste der aktuellen Parameter im Operandenfeld ist der Reihenfolge der Liste der formalen Argumente entsprechend aufgebaut (Stellungsparameter!), so daß eine entsprechende Substitution formaler Argumente durch aktuelle Parameter möglich ist. Die einzelnen Parameter müssen durch ein zulässiges Trennzeichen (Komma, Leerzeichen und/oder Tabulator) voneinander getrennt werden.

Beispiel:
Aufruf der Makrodefinition aus 4.1.1.:
START: PUT FELD,80.
Durch diesen Makroruf wird der Makro **PUT** aufgerufen, wobei das formale Argument **EABER** durch den aktuellen Parameter **FELD** und **LAENGE** durch **80.** ersetzt werden.

4.3. Die Substitution der Parameter

Die Substitution der formalen Argumente durch aktuelle Parameter erfolgt in der Reihenfolge ihrer Stellung (Stellungsparameter). Werden im Makroruf mehr Parameter als notwendig angegeben, so werden die nicht benötigten Parameter nicht berücksichtigt. Werden im Makroruf zu wenige Parameter angegeben oder Parameter ausgelassen, so werden die entsprechenden formalen Argumente als leer (*blank*) angenommen und bei der Makrogenerierung durch nichts ersetzt. Enthält ein aktueller Parameter Trennzeichen, so muß er in spitze Klammern eingeschlossen werden.

Beispiel:
BEISP FELD, ⟨MOV A,–(SP)⟩
Das erste formale Argument soll durch die Adresse **FELD** und das zweite durch den Befehl **MOV A,–(SP)** ersetzt werden. Da der zweite Parameter Trennzeichen (Leerzeichen und Komma) enthält, muß er in spitze Klammern eingeschlossen werden.

4.3.1. Die Parameterübergabe bei ineinander geschachtelten Makrorufen

In Makrodefinitionen können Makrorufe benutzt werden, wobei eine maximale Schachtelungstiefe von 16 nicht überschritten werden darf. In einem Makroruf, der sich innerhalb einer Makrodefinition befindet, können als aktuelle Parameter die formalen Argumente der Makrodefinition benutzt werden.

Beispiel:

```
.MACRO  MAC1 P1,P2,P3
...
...
MAC2    P2
...
.ENDM
```

Die Makrodefinition **MAC2** kann folgendermaßen definiert sein:

```
.MACRO  MAC2 P1,P2
...
...
.ENDM
```

Wenn der Makro **MAC1** wie folgt aufgerufen wird,
MAC1 #100,ALPHA, ⟨MOV R0,X0⟩
so wird der Makroruf
MAC2 ALPHA
generiert.

Unter Anwendung von spitzen Klammern können an den Makroruf **MAC2** auch mehrere Parameter übergeben werden.

Beispiel:
MAC1 #100, ⟨ALPHA,BETA⟩, ⟨MOV R0,X0⟩
Es wird der Makroruf
MAC2 ALPHA,BETA
generiert.

Die äußeren spitzen Klammern werden bei der Parameterübergabe nicht mit übergeben. Wenn die Notwendigkeit besteht, daß die spitzen Klammern mitgeneriert werden, so kann eine Klammerung mit der sogenannten Pfeilkonstruktion ^*x* ... *x* erfolgen. Durch den einstelligen Pfeiloperator wird das nachfolgende Zeichen *x* zum Trennzeichen zur Klammerung des Ausdruckes erklärt. Es kann ein beliebiges Zeichen sein, das im Klammerausdruck selbst nicht benutzt wird.

Beispiel:
Bei dem Makroruf
MAC1 #100,^&⟨MOV2(R5),C+6⟩, F0&,⟨MOV R0,X0⟩
wird der Makroruf
MAC2 ⟨MOV 2(R5),C+6⟩ ,F0
generiert.

Es besteht auch andererseits die Möglichkeit, das formale Argument der Makrodefinition, das als aktueller Parameter eines Makrorufes verwendet wird, in spitze Klammern oder in eine Pfeilkonstruktion einzuschließen, so daß die Übergabe der Klammern nicht unbedingt mit der Parameterübergabe organisiert werden muß.

Beispiel:

```
.MACRO  MAC1 P1,P2,P3
...
...
MAC2    ⟨P2⟩
...
.ENDM
```

Bei dem Makroruf
MAC1 #100,⟨MOV 2 (R5),C+6⟩,⟨MOV R0,X0⟩
wird der Makroruf
MAC2 ⟨MOV 2(R5),C+6⟩
generiert.

4.3.2. Die Verkettung der Parameter

In Makrodefinitionen ist der Apostroph **(')** als Trennzeichen zur Begrenzung der formalen Argumente zugelassen. Bei der Makrogene-

rierung wird der Apostroph als **Begrenzer** erkannt und entfernt, sofern er unmittelbar vor oder hinter einem formalen Argument steht. Diese Eigenschaft des Apostrophs wird genutzt, um aktuelle Parameter untereinander oder mit anderen Zeichen zu verketten. Wenn ein Apostroph erhalten bleiben soll, so sind in der Makrodefinition zwei Apostrophe zu schreiben.

Beispiel:
```
.MACRO MAC3 P1,P2,P3
P1'P3: .RAD50 /P1'P2'P3'0/
       .BYTE "P1,"P2
       .ENDM
```
Wenn der Makro durch folgende Quellanweisung aufgerufen wird:
MAC3 A,B,1Z2
so werden die Anweisungen
```
A1Z2: .RAD50 /AB1Z20/
      .BYTE 'A,'B
```
generiert.

4.3.3. Numerische Symbole

Zur Bildung von Symbolen in Makrodefinitionen können auch numerische Symbole benutzt werden. Dadurch wird es möglich, daß z. B. beim mehrmaligen Aufruf eines Makros doppelt definierte Symbole vermieden werden. Ein numerisches Symbol wird von einem aktuellen Parameter gebildet, wenn dem aktuellen Parameter der Operator \ (Backslash) im Makroruf vorangestellt wird. Der Operator \ bewirkt, daß der numerische Wert des aktuellen Parameters zum Zeitpunkt des Aufrufes des Makros in eine Zeichenkette entsprechend der aktuellen Basis des Zahlensystems umgewandelt wird, wobei Vornullen unterdrückt werden.

Beispiel:
```
.MACRO DEFW P1,P2
DEFWD  P1,P2, \MACIND
MACIND=MACIND+1

.ENDM
.MACRO DEFWD Z1,Z2,Z3
M'Z3:    .WORD Z1,Z2
         .ENDM
```
Bevor der Makro **DEFW** das erste Mal aufgerufen wird, muß **MACIND** definiert werden:
MACIND=0.
Beim ersten Aufruf des Makros
DEFW FELD,50.
werden die folgenden Anweisungen generiert:
```
m0:      .WORD FELD,50.
MACIND=MACIND+1
```
Es wird als Symbol **m0** generiert. Anschließend wird **MACIND** um 1 erhöht. Beim nächsten Aufruf des Makros **DEFW** wird dann das Symbol **M1** gebildet usw.

4.3.4. Die automatische Generierung lokaler Symbole

In Makrodefinitionen werden oft interne Symbole benötigt. Dafür können lokale Symbole benutzt werden. Der Makroassembler ersetzt formale Argumente automatisch durch lokale Symbole, wenn das formale Argument in der **.MACRO**-Anweisung durch ein vorangestelltes Fragezeichen (**?**) gekennzeichnet und im Makroruf das formale Argument durch keinen aktuellen Parameter ersetzt wird. Für die Makrogenerierung sind die lokalen Symbole **64**○ bis **127**○ reserviert. Es wird automatisch immer das nächste freie Symbol belegt. Wenn die lokalen Symbole erschöpft sind, so kann vom Programmierer durch eine **.ENABL-LSB**-Anweisung ein neuer lokaler Symbolblock eröffnet werden.

Beispiel:
```
.MACRO MAC4 P1,P2,?P3,?P4
MOV       P1,R1
P3:       CMP R0,(R1)+
          BMI P4
          DEC P2
          BNE P3
P4:
.ENDM
```
Durch den Makroruf
MAC4 FELD,ZAHL,,MARKE
werden folgende Anweisungen generiert:
```
MOV       #FELD,R1
64○:      CMP R0,(R1)+
          BMI MARKE
          DEC ZAHL
          BME 64○
MARKE:
```

4.4. Die Makroattributanweisungen

In der Makroassemblersprache MACRO-SM sind die Makroattributanweisungen **.NARG**, **.NCHR** und **.NTYPE** realisiert. Die Anweisung **.NARG** wird in Makrodefinitionen benutzt, um die Anzahl der im Makroruf spezifizierten aktuellen Parameter zu bestimmen. Die Anweisung hat folgende Syntax:

.NARG *symbol*

wobei dem Symbol ein Wert zugewiesen wird, der der Anzahl der aktuellen Parameter des Makrorufes entspricht.
Die Anweisung **.NCHR** kann in Assemblerprogrammen und in Makrodefinitionen benutzt werden, um die Anzahl der Zeichen einer Zeichenkette zu bestimmen. Besondere Bedeutung hat die Anweisung für die Bestimmung der Zeichenkettenlänge von aktuellen Parametern in Makrorufen. Sie hat folgende Syntax:

.NCHR *symbol[,⟨string⟩]*

wobei:
symbol – ein Symbol ist, dem ein Wert zugewiesen wird, der der Anzahl der Zeichen der Zeichenkette ⟨*string*⟩ entspricht;
, – ein zulässiges Trennzeichen (Komma, Leerzeichen und/oder Tabulator) ist;
⟨*string*⟩ – die Zeichenkette spezifiziert, von der die Länge bestimmt werden soll. Enthält die Zeichenkette Trennzeichen, so muß sie in spitze Klammern oder in eine Pfeilkonstruktion ^*x* ... *x* eingeschlossen werden. Die spitzen Klammern bzw. die Begrenzer der Pfeilkonstruktion werden nicht mitgezählt. Wenn die Zeichenkette spitze Klammern enthält, so muß die Zeichenkette mit einer Pfeilkonstruktion eingeschlossen werden. Wird in der Anweisung keine Zeichenkette spezifiziert, so wird dem Symbol der Wert 0 zugewiesen.

Beispiel:
```
.NCHR  ZZ,⟨ALPHA=⟩
              ;DEM SYMBOL ZZ WIRD
              ;DER WERT 6 ZUGEWIE-
              ;SEN
NCHR   X0,^/PC UND SP ⟨CR⟩/
              ;DEM SYMBOL ZZ WIRD
              ;DER WERT 15 ZUGEWIE-
              ;SEN
```

Die Anweisung **.NTYPE** wird in Makrodefinitionen benutzt, um den Adressierungsmodus eines aktuellen Parameters eines Makrorufes zu bestimmen. Sie hat folgenden Aufbau:

.NTYPE *symbol[,expr]*

wobei:
symbol – ein Symbol ist, dem ein Wert zugewiesen wird, der dem Adressierungsmodus des Adreßausdrucks entsprechend Tafel 8 charakterisiert ist (6 Bit);
, – ein zulässiges Trennzeichen (Komma, Tabulator und/oder Leerzeichen) ist;
expr – ein zulässiger Adreßausdruck ist, wie er auch in symbolischen Maschinenbefehlen benutzt werden kann. Wenn kein Ausdruck spezifiziert wird, so wird dem Symbol der Wert 0 zugewiesen.

Beispiel:
```
.MACRO LAD P1
.NTYPE MOD,P1
.IF    EQ,MOD-27
MOV    P1,R4
.MEXIT
.ENDC
.IF    NE,MOD-67
.ERROR ;FALSCHER ADRESSIE-
       ;RUNGSMODUS
.MEXIT
.ENDC
MOV    #P1,R4
.ENDM
```
Bei dem Makroruf
LAD #FELD+2
wird der Befehl MOV #FELD+2,R4 generiert.
Bei dem Makroruf
LAD FELD+2
wird der gleiche Befehl wie beim vorhergehenden Aufruf generiert. Bei allen anderen Adressierungsmodi wird kein Befehl generiert und die Generierung mit einem P-Fehler abgebrochen.

4.5. Makrodefinitionen aus der Makrobibliothek

Für den Anwender besteht die Möglichkeit, Makrodefinitionen mit dem Bibliothekar LBR in einer Makrobibliothek abzuspeichern. Außerdem kann der Nutzer auf die Systemmakrobibliothek zurückgreifen. Der Aufruf der Makrodefinitionen aus den Makrobibliotheken erfolgt mit der **.MCALL**-Anweisung vor der ersten Generierung. Sie hat folgende Syntax:

.MCALL *arg [, arg, [, ...]]*

Tafel 8 Übersicht über die Adressierungsarten

Notation	Numerischer Wert	Bezeichnung	Bedeutung
R	0n	Registeradressierung	Das Register R enthält den Operanden.
(ER) oder @R	1n	Register-Indirekt-Adressierung	Das Register R enthält die Adresse des Operanden.
(ER)+	2n	Autoinkrement-Adressierung	Das Register, spezifiziert durch ER, enthält die Adresse des Operanden. Nach der Operation wird der Registerinhalt erhöht.
@(ER)+	3n	Autoinkrement-Indirekt-Adressierung	Das Register, spezifiziert durch ER, enthält die Adresse der Adresse des Operanden. Nach der Operation wird der Registerinhalt erhöht.
−(ER)	4n	Autodekrement-Adressierung	Das Register, spezifiziert durch ER, enthält die Adresse des Operanden, die vor der Operation erniedrigt wird.
@−(ER)	5n	Autodekrement-Indirekt-Adressierung	Das Register, spezifiziert durch ER, enthält die Adresse der Adresse des Operanden. Vor der Operation wird der Registerinhalt erniedrigt.
E(ER)	6n	Index-Adressierung	Der Ausdruck E plus der Inhalt des Registers, spezifiziert durch ER, ergeben die Adresse des Operanden. Die Indexkonstante E steht nach dem OPC im Speicher.
@E(ER)	7n	Index-Indirekt-Adressierung	Der Ausdruck E plus der Inhalt des Registers, spezifiziert durch ER, ergeben die Adresse der Adresse des Operanden. Die Indexkonstante E steht nach dem OPC im Speicher.
#E	27	Direktwertadressierung	Der Ausdruck E ist der Operand selbst. Der Direktwert steht nach dem OPC im Speicher.
@#E	37	Absolute Adressierung	Der Ausdruck E ist die Adresse des Operanden. Nach dem OPC steht die absolute Adresse im Speicher.
E	67	Relative Adressierung	Der Ausdruck E ist die Adresse des Operanden. Nach dem OPC steht die relative Adresse als Differenz der Operandenadresse und des Befehlszählers im Speicher.
@E	77	Indirekte relative Adressierung	Der Ausdruck E ist die Adresse der Adresse des Operanden. Nach dem OPC steht die relative Adresse als Differenz der Adresse der Adresskonstante und des Befehlszählers im Speicher

n – Natürliche Zahl zwischen 0 und 7 als Registernummer
R – Registerausdruck (Register expression)
E – Ausdruck (Expression)
ER – Registerausdruck oder Ausdruck, der einen Wert von 0 bis 7 ergibt.

wobei *arg* für die Namen der benötigten Makrodefinitionen steht. Die Suche der Makrodefinitionen erfolgt zunächst in den Nutzerbibliotheken, die in der Kommandozeile für den Makroassembler spezifiziert wurden. Danach wird die Suche in der Systemmakrobibliothek fortgesetzt. Wird eine Makrodefinition nicht gefunden, so wird das Fehlerzeichen U gesetzt.

4.6. *Indirekte Makrodefinitionen*

Unter einer indirekten Makrodefinition wird die Definition eines Wiederholungsblockes verstanden, die eine zyklische Generierung der Anweisungen des Wiederholungsblockes an der Stelle im Programm bewirkt, wo der Wiederholungsblock definiert ist. Wiederholungsblöcke können im Assemblerprogramm und in Makrodefinitionen angewendet werden.
Wiederholungsblöcke werden durch eine **.IRP**-, **.IRPC**- oder **.REPT**-Anweisung eingeleitet und mit einer **.ENDM**-Anweisung abgeschlossen. Für die Substitution formaler Argumente durch aktuelle Parameter gelten die gleichen Regeln wie für Makrodefinitionen (siehe 4.3.).

4.6.1. Die .IRP-Anweisung

Bei der **.IRP**-Anweisung wird ein formales Argument nacheinander durch mehrere aktuelle Parameter ersetzt. Dabei werden die Modellanweisungen des Wiederholungsblockes für jeden aktuellen Parameter einmal generiert. Die Anzahl der Wiederholungen wird somit durch die Anzahl der aktuellen Parameter bestimmt. Der Wiederholungsblock mit der **.IRP**-Anweisung hat folgenden Aufbau:

.IRP *arg*, ⟨*par1*,*par2*,...⟩
...
...
Modellanweisungen
...
...
.ENDM

wobei
arg – ein Symbol darstellt, das als formales Argument benutzt wird;
, – ein zulässiges Trennzeichen (Komma, Tabulator und/oder Leerzeichen) ist;
⟨*par1*,*par2*, ...⟩ – die Liste der aktuellen Parameter darstellt. Die aktuellen Parameter müssen durch ein zulässiges Trennzeichen getrennt und in spitze Klammern eingeschlossen werden. Die einzelnen Parameter können bei Notwendigkeit selbst in spitze Klammern oder in eine Pfeilkonstruktion eingeschlossen werden.

Beispiel:
```
.IRP  P1, ⟨ALPHA,BETA,GAMMA⟩
 ADD  P1,R0
.ENDM
```
Es werden folgende Anweisungen generiert:
```
 ADD  ALPHA,R0
 ADD  BETA,R0
 ADD  GAMMA,R0
```

4.6.2. Die .IRPC-Anweisung

Bei der **.IRPC**-Anweisung wird ein formales Argument nacheinander durch das jeweils nächste Zeichen einer Zeichenkette ersetzt. Die Anzahl der Generierungen des Wiederholungsblockes wird durch die Länge der Zeichenkette bestimmt. Der Wiederholungsblock mit der **.IRPC**-Anweisung hat folgenden Aufbau:

.IRPC *arg*, ⟨*string*⟩
...
...
Modellanweisungen
...
.ENDM

wobei
arg – ein Symbol darstellt, das als formales Argument benutzt wird;
, – ein zulässiges Trennzeichen (Komma, Tabulator und/oder Leerzeichen) ist;
⟨*string*⟩ – die Zeichenkette darstellt, deren Zeichen als aktuelle Parameter eingesetzt werden. Die Zeichenkette muß in spitze Klammern eingeschlossen werden, wenn sie Trennzeichen enthält.

Beispiel:
```
.IRPC  P0,ERG=
 MOVB#"P0,(R2)+
.ENDM
```
Es werden folgende Anweisungen generiert:
```
MOVB  #'E,(R2)+
MOVB  #'R,(R2)+
MOVB  #'G,(R2)+
MOVB  #'=,(R2)+
```

4.6.3. Die .REPT-Anweisung

Die **.REPT**-Anweisung wird benutzt, um eine Anzahl von Quellanweisungen mehrfach unverändert zu generieren. Der Wiederholungsblock mit der **.REPT**-Anweisung hat folgenden Aufbau:

.REPT *expr*
...
Quellanweisungen
...
.ENDM (auch **.ENDR**)

wobei
expr – ein zulässiger Ausdruck ist, der die Anzahl der Wiederholungen angibt. Wenn der Ausdruck den Wert 0 ergibt, so wird keine Anweisung generiert.

(wird fortgesetzt)

32-Bit-Mikroprozessoren

Wolf-Dietram Bretschneider, Dresden

Die derzeitigen Integrationsgrade ermöglichen die Herstellung von Mikroprozessoren, die den sogenannten Minicomputern ähneln und aufgrund der 32-Bit-Verarbeitungsbreite bereits Aufgaben lösen, die bisher Großrechnern vorbehalten waren. Dabei existiert meist der Prozessor nicht allein auf dem Chip, sondern es sind Einheiten wie Fließkommaprozessor (FPU – Floating Point Unit) oder Speicherverwaltungseinheit (MMU – Memory Management Unit) mit integriert. Infolge der Aufwärtskompatibilität können die Programme der kleineren „Verwandten" einer Prozessorfamilie weiterhin verwendet werden. Im folgenden Text werden einige Vertreter der 32-Bit-Technik vorgestellt.

1. Allgemeine Merkmale

Hervorzuheben sind die fein abgestuften Unterbrechungsstrukturen, die Unterstützung von Multitaskingsystemen und vielfältige Debuggerfunktionen zur Erleichterung der Programmentwicklung. Außerdem besitzen alle Prozessoren spezielle Befehle zur Kommunikation mit Entlastungsprozessoren (EPU). Neben den bekannten Datentypen wie Byte, Wort oder Langwort werden gepackte und ungepackte BCD-Zahlen, Quadwort-Daten mit 64 Bit Länge und Bitfelder mit angegebener Feldweite w verarbeitet. Als Bitfeld versteht man eine Reihe, bestehend aus einer bestimmten Anzahl Bits, die an einer vorgegebenen Adresse beginnt. Zu dieser Zugriffsadresse gehört noch ein Offset, der die Anzahl der Bits bis zum ersten Bit des Feldes angibt. Bild 1 zeigt die Datenorganisation.

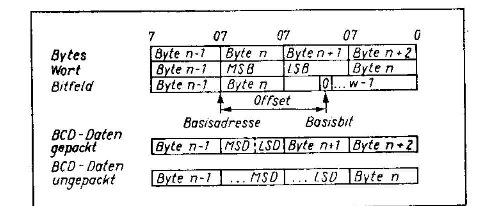


Bild 1 Datenformate im Rechner

Ein weiteres Merkmal der 32-Bit-Prozessoren ist das gleichzeitige Laden der nächsten Befehle in mehrstufige Register (Pipeline) und das Abarbeiten vorheriger Instruktionen. Besonders bewähren sich dabei auf dem Prozessorchip integrierte Cache-Speicher, die einen extrem schnellen Arbeitsspeicher vortäuschen. Findet der Prozessor eine gültige Information im Cache, so entfällt der Buszyklus zum Laden. Andernfalls wird bei Ausführung des Speicherzugriffs auch der Cache aktualisiert.

Die Buszugriffe erfolgen asynchron, das heißt, der Prozessor erhält vom angesprochenen Speicher ein Signal, wenn sich gültige Daten auf dem Bus befinden. Zum Auffinden von Softwarefehlern stehen spezielle Bits im Statusregister zur Verfügung, die ein schrittweises Abarbeiten der Befehle (Trace) bzw. ein Bearbeiten des Programms bedingungsgesteuert (Trace-Pending) zulassen. Außerdem besteht die Möglichkeit zum Setzen von Unterbrechungspunkten im Programmablauf.

In Tafel 1 ist eine Übersicht einiger 32-Bit-Prozessorfamilien dargestellt.

Tafel 1 Übersicht einiger 32-Bit-Familien

Hersteller	Bezeichnung	spezielle Eigenschaften
National Semiconductor	NS 32008	8-Bit-Datenbus, 24-Bit-Adreßbus, intern 32 Bit
	NS 32016	16-Bit-Datenbus, sonst wie 32008
	NS 32032	32-Bit-Daten- und Adreßbus
Zilog	Z 80000	32-Bit-Daten- und Adreßbus intern 32 Bit, mit MMU
Intel	I 80386	32-Bit-Daten- und Adreßbus große virtuelle Speicher adressierbar, mit MMU
Motorola	MC 68008	8-Bit-Datenbus
	MC 68000	16-Bit-Datenbus, 24-Bit-Adreßbus, intern 32 Bit
	MC 68010	Weiterentwicklung des 68000 mit Vektor-Basis-Register
	MC 68020	32-Bit-Daten- und Adreßbus Modulkonzept
	MC 68030	Weiterentwicklung des 68020 mit doppelter Cachestruktur für Daten und Befehle, mit MMU

2. National Semiconductor 32032

Der 32032 war bzw. ist einer der ersten 32-Bit-Prozessoren mit guter Unterstützung höherer Programmiersprachen und mit Modulkonzepten. In Bild 2 wird das Programmiermodell dargestellt.

Der 32032 kann 16 MByte adressieren, enthält 8 Universalregister, 32 Bit breit, zum Arbeiten mit den Operanden. Weiterhin stehen 24-Bit-Register zur indizierten oder relativen Adressierung, zwei Stackpointer zur Trennung von System- und Anwenderstack sowie zusätzliche Register für Flags und den aktuellen Prozessor- und Modulstatus zur Verfügung. Bei einem Modulaufruf enthält das Register MOD eine genaue Beschreibung des Moduls. Das 4-Bit-Register CFG gibt Aufschluß über die Rechnerkonfiguration mit eventuell vorhandenen FPU (Bit = 1), MMU (Bit M= 1), EPU (Bit C = 1) oder externem Interruptcontroller (Bit I = 1).

3. Intel 80386

Die in Bild 3 dargestellten Register erinnern stark an die Vorgängermodelle des 80386 (z. B. Register EAX = AX-Reg. des 8086), zu denen er codekompatibel ist. Aufgrund reich-

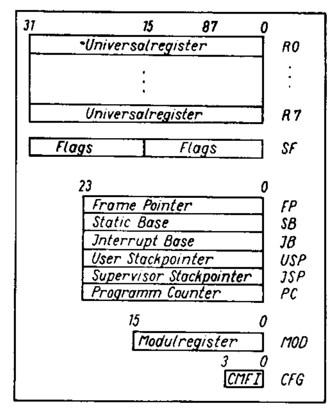


Bild 2 Registersatz des 32032

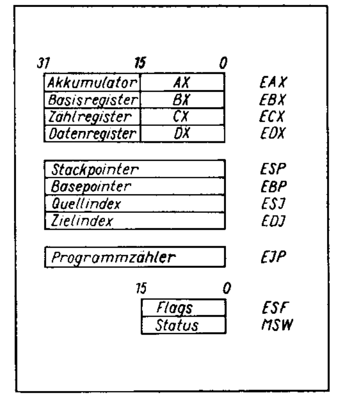


Bild 3 Register des 80386 ohne MMU

Bild 4 Register des Z 80000 ohne MMU

Bild 5 Register des 68020 (68030 ohne MMU)

Wolf-Dietram Bretschneider (33) absolvierte von 1978 bis 1983 ein Elektronikfernstudium an der Ingenieurschule für Elektronik und Informationsverarbeitung Görlitz und von 1985 bis 1987 ein postgraduales Studium Mikroprozessortechnik an der Technischen Universität Karl-Marx-Stadt.
Er ist gegenwärtig als Prüffeldingenieur für rechnergesteuerte Prüfautomaten tätig.

lich vorhandener Software gehört die Intelfamilie zu den am weitesten verbreiteten Mikroprozessoren. Im virtuellen 8086-Modus kann der 80386 uneingeschränkt Vorgängerprogramme abarbeiten. Beim Übergang von Mehrrechnersystemen mit dem 8086 zum Einsatz eines 80386 in dem System sind keine Programmänderungen nötig.
Die Register EAX ... EDX ermöglichen das Arbeiten mit Operanden, die Zeiger- und Indexregister dienen zusammen mit den nicht im Bild angegebenen Segment-Selektor-Registern der mit auf dem Chip integrierten MMU der Adreßmanipulation. Die Leistungsfähigkeit der MMU garantiert die sichere Verwaltung des virtuellen Speichers. Im Maschinen-Status-Wort (MSW) werden der Adreßmode und die Funktion der MMU vereinbart. Das Flag-Control-Wort unterteilt sich in System- und Anwenderbyte.

4. Zilog Z 80000

Wie schon der Z 8000, so fällt auch sein Nachfolger durch die ungewöhnlich hohe Anzahl von Universalregistern auf, deren Länge auf 64 Bit erweitert werden kann. Der Z 80000 hat ebenfalls die MMU mit auf dem Prozessorchip integriert und besitzt einen 256 Byte großen Cachespeicher. In Bild 4 wird die Architektur verdeutlicht.
Die CPU kann in drei Adreßmodes arbeiten, die im Flag-Control-Wort vereinbart werden. Dabei entspricht der Compactmode dem Z 8002 mit 64 KByte adressierbarem Speicherbereich. Der segmentierte Modus unterscheidet Segmente mit einer Größe von 64 KByte (Z 8001) und 16 MByte. Im linearen Modus ist die Adresse 32 Bit lang. Die Unterscheidung von Normal- und Systemmode bedingt getrennte Stackbereiche. Dazu existieren der Normalstackpointer als separates 32-Bit-Register und die vom Adreßmode abhängige Verwendung der Register RR12 und RR13 als Systemstack- bzw. Framepointer. Bei schwerwiegenden Systemfehlern werden die zur Wiederherstellung der CPU-Funktion benötigten Informationen aus dem Speicherbereich geladen, den der OSP adressiert. Das HICR beschreibt Merkmale der Hardwarekonfiguration im Zusammenspiel mit der CPU (z. B. Wartezyklen für Speicher- und Peripherieschaltkreise oder Angaben über die Portweiten bei systemfremder Peripherie).

5. Motorola 68020

Der 68000 der Firma Motorola war der erste Prozessor, der mit 16-Bit-Daten- und 24-Bit-Adreßbus intern über 32 Bit breite Register verfügte. Die ständige Weiterentwicklung des Grundtyps führte über den 68010 zum echten 32-Bit-Prozessor 68020 bzw. 68030. Die Registerstruktur stellt Bild 5 dar.
Das Vektor-Basis-Register enthält die Anfangsadresse der Sprungtabelle bei Ausnahmebehandlungen während des Programmablaufes und wurde erst ab dem 68010 implementiert. Diese erste Weiterentwicklung zeichnete sich durch eine Vervollkommnung des sogenannten Loop-Modus aus, der veranlaßt, daß die dafür vorgesehenen Befehle innerhalb bedingungsgesteuerter Schleifen in dem mehrstufigen Befehlsregister verbleiben. Es werden nur noch für die zu verarbeitenden Daten Buszyklen ausgeführt. Die Funktionscodeleitungen lassen neben der Interrupt- bzw. Breakpointbestätigung eine Unterscheidung der angesprochenen Speicherbereiche zu. Die unbenutzten Codierungen können mittels spezieller Befehle vom Programmierer in die Alternativ-Funktions-Coderegister geladen werden und zum Schreiben (SFC) oder Lesen (DFC) in fremden Speicherbereichen benutzt werden, da sie als extra Status ausgegeben sind. Ab dem 68020 mit integriertem Cache für Befehle gibt es zwei weitere Register. Das CAAR bestimmt das Feld des Cachespeichers, das jeweils durch einen speziellen vom CACR ausgelösten Befehl gelöscht wird. Beim Betrachten der Daten- und Adreßregister fällt die Dreiteilung des als Stackpointer verwendeten A7 auf.
Welcher der beiden Stackpointer im Supervisormode aktiv ist, A7' oder A", bestimmt ein Bit des Statusbytes. Die Umschaltung ist privilegiert, und falls ein Bit während des Programmlaufes nicht verändert wird, bleibt der 68020 kompatibel zu seinen Vorgängern. Die Dreiteilung läßt das Ablegen aller residenten Prozeßdaten auf dem Masterstack zu, während auf dem Interruptstack der normale Supervisorbetrieb abläuft. Soll ein Taskwechsel stattfinden, so wird vom Betriebssystem der momentane Status der abgeschalteten Task im Masterstack abgelegt. Diese Vorgehensweise findet man auch bei der Weiterentwicklung, dem 68030, der nebem einer MMU einen zweiten Cachespeicher für Daten auf dem Chip integriert hat. Durch das intern doppelt ausgelegte Bussystem können Cache und MMU parallel arbeiten.

Literatur

/1/ Stuhlmüller, P.: 16 Bit Generation Z 8000. München te-wi-Verlag 1980
/2/ Grohmann, B.; Eichler, L.: Das Prozessorbuch zum 68000. Data Becker Verlag, Düsseldorf 1985
/3/ Fäh, P.: CALM-Allgemeine Assemblersprache. Fribourg/Schweiz 1985
/4/ Nachtmann, L: Neue Mikroprozessoren. Chip 12/1986
/5/ Z 80000 Produkt Spezifikation. Zilog 1983

Technik international

heißt eine neue MP-Rubrik, in der wir in loser Folge in Form eines „Steckbriefs" internationale Neuentwicklungen vorstellen.

32-Bit-PC PCD-3T

Der Siemens PCD-3T ist ein Standmodell auf Basis des 32-Bit-Prozessors 80386. Er erweitert die Produktpalette der Siemens-PC nach oben hin. Eingesetzt wird er als leistungsfähiger Server im Netz oder an Abteilungsarbeitsplätzen. Der PCD-3T verfügt über 6 freie Steckplätze für Erweiterungskarten. Es können bis zu 7 Slimline-Laufwerke eingebaut werden. Als Diskettenlaufwerke werden 3½" sowie 5¼" angeboten, die auf Wunsch auch im Gerät kombiniert werden können. Neben MS-DOS wird 1988 als neues Betriebssystem auch MS-OS/2 angeboten werden.
Die Bildschirmdarstellung ist schwarz auf weiß. Auf inverse Darstellung kann umgeschaltet werden. Es ist auch ein Farbbildschirm anschließbar.
Der PCD-3T enthält eine Elektronikgrundplatine mit dem Prozessor 80386, der mit einer Taktfrequenz von 16 MHz betrieben wird. Prozessor und Speicher arbeiten durch spezielle Technik so eng zusammen, daß die meist üblichen Wartezyklen bei Speicheroperationen entfallen können. Damit wird die Leistung des Prozessors bestmöglich genutzt. Zur Beschleunigung von arithmetischen Rechenoperationen kann der Numerikprozessor 80387 eingebaut werden. Die Taktfrequenz kann auf 6 MHz oder 8 MHz zurückgeschaltet werden. Der Speicherausbau beginnt bei 1 MByte und kann auf der Grundplatine integriert bis auf 16 MByte erweitert werden.
Zur Softwareausstattung des PCD-3T gehören MS-DOS 3.2, MS-Windows mit MS-Write und MS-Paint sowie GW-Basic. *MP*

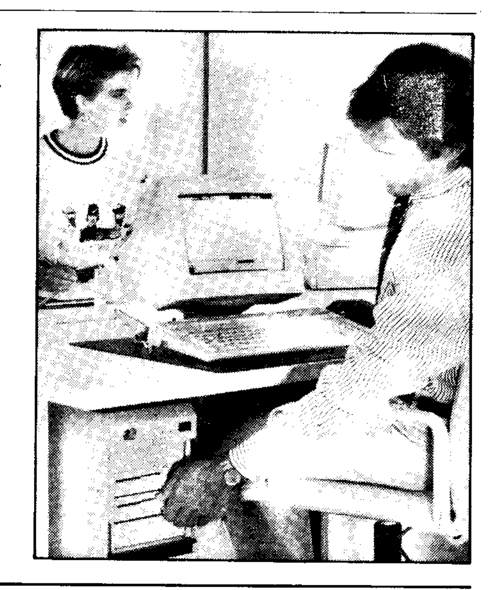


Foto: Siemens

256-K-dRAM-Modul für KC 85/2 (/3)

Albrecht Barthel, Holger Krieg
Strausberg

Grundlagen

Ausgehend vom Systemkonzept der Kleincomputer aus Mühlhausen, insbesondere der Möglichkeit der softwaremäßigen Verwaltung der Speichermodule, ist es machbar, Speicher der Größenordnung 256 KByte und mehr eineindeutig zu verwalten. Die Adressenerzeugung erfolgt über OUT-Befehle der CPU, wobei durch die Auswertung des Datenbusses die Adresse eingestellt wird. So ist es auch möglich, daß Speicherblocks eines Moduls in ihrer Basisadresse verschoben werden können. Beispielsweise kann der Modul M022 (16 KdRAM) auf vier verschiedene Basisadressen (0000H), 4000H, 8000H, C000H) gelegt werden. Analog ist es beim 64-K-dRAM-Modul M011, wobei dann die Speicherblocks (jeweils 16 K) innerhalb des 64-K-Adreßraumes entsprechend verschoben werden. Die Tafel 1 soll dies darstellen. Sie wurde /1/ entnommen.
Zur Realisierung dieser Verschiebung (Generierung der Basisadresse) werden die Bits 6 und 7 des Datenbusses ausgewertet. Zur Aktivschaltung wird das Bit 0 ausgewertet. Das Bit 1 dient als Schreibschutzbit und ist low-aktiv. Die Bits 2, 3, 4 und 5 bleiben also beim 64-K-Modul unausgewertet. Auf der Basis der Leiterkarte des Moduls M011 (64-K-RAM) wurde mit den Schaltkreisen 41256 (Organisation 256Kx1 Bit) ein 256-K-dRAM-Modul aufgebaut. Zur Verwaltung dieses Moduls werden im Steuerbyte zusätzlich die Bits 2 und 3 verwendet (Tafel 2).
Durch Auswerten der Bits 2 und 3 ist es somit möglich, vier 64-K-Blöcke zu selektieren. Es sind damit vier Ebenen (Blocks) im Modul mit jeweils 64 KByte verfügbar. Auf diese 64 KByte kann die CPU entsprechend der Prioritätslogik zugreifen, d. h., es darf, um zugreifen zu können, kein höherpriorisierter Speicherbereich aktiv auf dem gleichen Adreßbereich liegen. Jede dieser Ebenen verhält sich somit genauso im System wie ein 64-K-Modul. Siehe dazu auch /1/ und /2/.

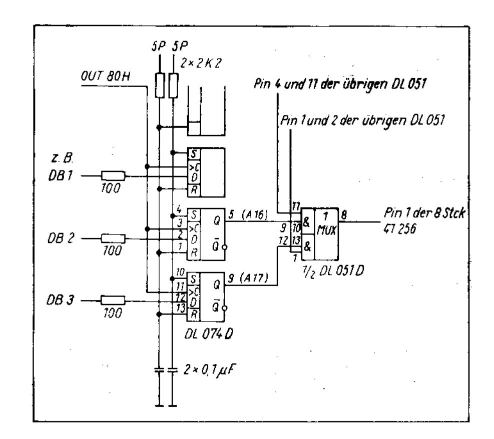


Bild 1 Schaltplan (Auszug)

Tafel 2 Belegung des Steuerbytes

Bit 0	Modulaktivbit 0 – Modul inaktiv 1 – Modul aktiv
Bit 1	Schreibschutzbit 0 – nur Lesen 1 – Lesen und Schreiben
Bit 2	Ebenenselekt (1)
Bit 3	Ebenenselekt (1) 00 – Ebene 0 01 – Ebene 1 10 – Ebene 2 11 – Ebene 3
Bit 4	bleibt unausgewertet
Bit 5	bleibt unausgewertet
Bit 6	Basisadresse
Bit 7	Basisadresse 00 – Basisadresse 0000H 01 – Basisadresse 4000H 10 – Basisadresse 8000H 11 – Basisadresse C000H

(1) diese Bits werden nur beim 256-K-Modul ausgewertet

Tafel 1 Basisadressen des 64-K-Adreßraumes

Basisadresse	Steuerbyte	Adr. 0000	Adr. 4000	Adr. 8000	Adr. C000
0000 H	03	Block 1	Block 2	Block 3	Block 4
4000 H	43	Block 2	Block 1	Block 4	Block 3
8000 H	83	Block 3	Block 4	Block 1	Block 2
C000 H	C3	Block 4	Block 3	Block 2	Block 1

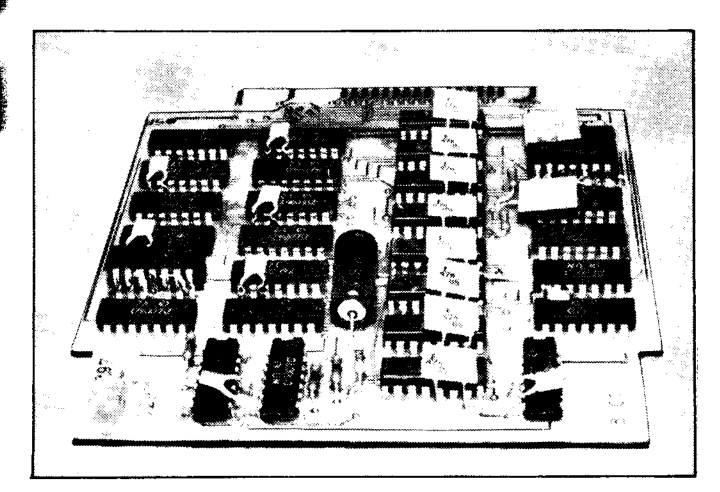
Hardware

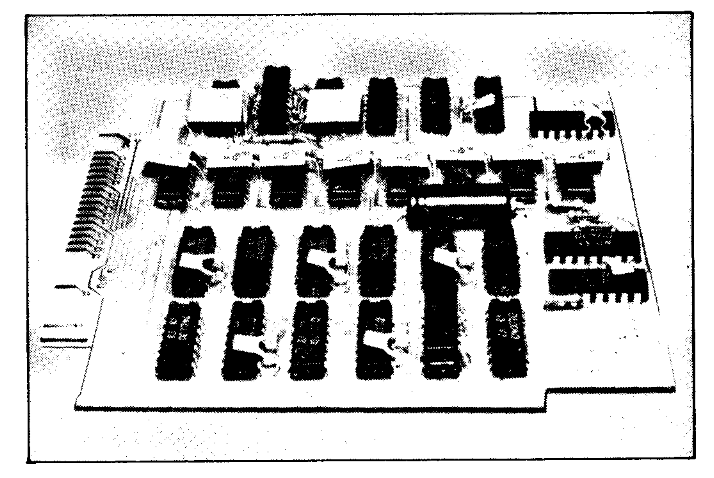
Wie bereits beschrieben, werden die Bits 2 und 3 ausgewertet. Diese Auswertung erfolgt, indem die Datenleitungen der Bits 2 und 3 über einen Widerstand (100 Ohm) mit den Dateneingängen je eines R-S-Flip-Flops (DL 074) verbunden werden. Über die OUT-Adresse 80 H wird dieses Flip-Flop getaktet. D. h., bei einem OUT-Befehl mit der OUT-Adresse 80H werden die Ausgänge der beiden Flip-Flops dem Logikpegel der Datenbits 2 und 3 entsprechend gesetzt. Die Flip-Flop-Ausgänge werden einem Multiplex-Gatter eines zusätzlichen Schaltkreises DL 051 zugeführt. Am Ausgang dieses Multiplexers steht dabei die Adresse A8 zur Verfügung. Diese Adresse wird dem Pin 1 der acht Schaltkreise 41 256 bzw. dem pinkompatiblen sowjetischen Typ K565RU7 (256K∗1 Bit-dRAM mit 7-Bit-Refresh) zugeführt und entspricht den Adreßbits A 16 und A 17. Das beschriebene Prinzip wird im Schaltplan (Bild 1) nochmal konkret dargestellt. Den fertigen Modul, entstanden aus dem Modul M011 (64 KdRAM) zeigen die Fotos (Bilder 2 und 3).
Die Modulsteuerworterzeugung wird über relativ dressierte OUT-Befehle der CPU realisiert. Dabei haben die einzelnen Register folgende Funktionen:

Register C:
Der Inhalt des Registers C bildet die OUT-Adresse, nach der ein OUT-Befehl gegeben wird. Das wird auch als relative Adressierung bezeichnet.

Register B:
Bei relativ adressierten OUT-Befehlen wird der Inhalt des Registers B auf die Adreßbits A8 und A15 gelegt. Diese Eigenschaft wird beim Systemkonzept des HC900 und Nachfolgemodellen konsequent genutzt, indem der Inhalt des Registers B die Steckplatzadresse des Moduls (08H oder 0CH) bildet.

Akkumulator (Register A):
Der Inhalt des Akku wird beim entsprechen-





Bilder 2 und 3 Der fertige Modul aus unterschiedlicher Perspektive gesehen *Fotos (2): Barthel*

den OUT-Befehl auf den Datenbus gelegt und wie oben beschrieben ausgewertet. Der OUT-Befehl hat also die Form:
OUT (C), A
Diese Methodik wird u. a. bei dem Systemprogramm SWITCH bzw. dem Systemunterprogramm MODU (UP Nr.: 26H) konsequent genutzt. Siehe dazu auch /2/.

Zur Software

Bis zur Verfügbarkeit des Erweiterungsaufsatzes zur Ansteuerung eines oder mehrerer Floppy-Disk-Laufwerke kann der vorgestellte Modul sehr effektiv unter dem derzeitigen ROM-residenten Betriebssystem CAOS verwaltet werden. Dazu wurden bisher vier Menüfunktionen EBENE, INIT, SYs und RSYs implementiert.
Dieser kleine Zusatz zum Monitor residiert ab BE00H. Es wird damit folgendes realisiert:
Das Programm ist auf Kassette als selbststartendes Maschinenprogramm abgelegt und löscht unmittelbar nach erfolgreichem Laden alle 4 Ebenen des RAM-Moduls vollständig. Erst dann können die aufgeführten Kommandos aufgerufen werden. Die Kommandos INIT, SYs und RSYs sind im Menü nicht sichtbar, können aber aufgerufen werden. Siehe dazu auch /3/.

Funktion EBENE (n)

Es wird als Parameter die gewünschte Ebene aktiviert. Als Parameter sind also die Ziffern 0 bis 3 zulässig. Durch das Programm wird dann folgender Algorithmus abgearbeitet:

- Verlegen des System-Workspace in den IRM ab BFFFH abwärts
- Zuschalten des 256-K-RAM-Moduls in der entsprechenden Ebene
- Abschalten des internen RAM
- Rückverlegen der System-Zellen auf 0130 bis 0200H.

Der Speicher wird bei Aufruf des Kommandos EBENE nicht gelöscht, das heißt, es

4a

```
6000              ;***************************************
6000              ;
6000              ;         PROGRAMMNAME:
6000              ;                            SOFT9
6000              ;
6000              ;         STAND:             17.06.1987
6000              ;         (c)                A.Barthel
6000              ;
6000              ;---------------------------------------
6000              ;
6000              ;         EBENE     n (0 ... 3)
6000              ;                   n gerufene Ebene
6000              ;                   ohne Parameter nur
6000              ;                   anzeigen !
6000              ;
6000              ;         SYs       legt System in den IRM
6000              ;         RSYs      legt System in den RAM
6000              ;
6000              ;***************************************
6000              ;
6000                        ORG       0BE00H
BE00              ;
BE00              ;
BE00              PV1       EQU       0F003H
BE00              PV3       EQU       0F009H
BE00              REST      EQU       0E000H
BE00              ARGN      EQU       0B781H
BE00              STCK      EQU       0B7AEH
BE00              ;
BE00              ;         es folgen Arbeitszellen,die im
BE00              ;         RAM liegen muessen
BE00              ;         im IRM liegen sie am Ende des
BE00              ;         Funktionstastenspeichers,deshalb
BE00              ;         Vorsicht bei der Arbeit damit !
BE00              ;
BE00              MSCH      EQU       0B99DH ;MERKER SCHACHT
BE00              MAE       EQU       0B99EH ;MERKER AKT.EBEN
BE00              MGE       EQU       0B99FH ;MERKER GER.EBEN
BE00              ;
BE00              ;
BE00              ;
BE00              ;
BE00 C30ABE                 JP        INIT      ;SELBSTSTARTADR.
BE03              ;
BE03              ;---------------------------------------
BE03              ;         INIt
BE03              ;---------------------------------------
BE03              ;
BE03 7F7F                   DEFW      7F7FH     ;PROLOG
BE05 494E4974               DEFM      'INIt'
BE09 01                     DEFB      1
BE0A F3           INIT      DI                  ;INT AUS
BE0B CDC0BE                 CALL      TST       ;STECKPLATZ SUCH
BE0E 2114BE                 LD        HL,RIN1   ;RUECKKEHRADR.
BE11 C329BF                 JP        IIRM      ;SYST->IRM
BE14 CD47BF       RIN1      CALL      RAMA      ;SYST.RAM AUS
BE17 3A9DB9                 LD        A,(MSCH);STCKPL.->L
BE1A 6F                     LD        L,A
BE1B 1603                   LD        D,3       ;EBENE 0, L&S
BE1D CDDDBE                 CALL      MCRS      ;4*32K->0
BE20 CBFA                   SET       7,D       ;OBERE 32K
BE22 CDDDBE                 CALL      MCRS      ;->0
BE25 CD4CBF                 CALL      RAME      ;SYST.RAM EIN
BE28 CD3FBF                 CALL      MODAU     ;MODUL AUS
BE2B 3E03                   LD        A,3
```

4b

```
BE2D 329EB9                 LD        (MAE),A ;INIT.EBENENMERK
BE30 2168BF                 LD        HL,EXH
BE33 C330BF                 JP        IRAM      ;SYSTEM ZURUECK
BE36              ;
BE36              ;---------------------------------------
BE36              ;         EBENE
BE36              ;---------------------------------------
BE36              ;
BE36 7F7F                   DEFW      07F7FH
BE38 4542454E               DEFM      'EBENE'
BE3D 01                     DEFB      1
BE3E E5                     PUSH      HL        ;PARAMETER RETT.
BE3F 3A81B7                 LD        A,(ARGN)
BE42 FE00                   CP        0         ;KEIN PARAMETER?
BE44 CA7DBE                 JP        Z,EBEN1 ;NUR ANZEIGEN
BE47 E1                     POP       HL
BE48 F3                     DI                  ;INT AUS !
BE49 7D                     LD        A,L
BE4A CDB0BE                 CALL      PARTST
BE4D 329FB9                 LD        (MGE),A;STEUERBYTE L&S
BE50 CDC0BE                 CALL      TST       ;KANN BEI
BE53 2159BE                 LD        HL,RT1    ;SELBSTST. ENTF.
BE56 C329BF                 JP        IIRM      ;SYST.->IRM
BE59 3A9EB9       RT1       LD        A,(MAE)
BE5C CD1EBF                 CALL      MVOR      ;EINSCH&VERSCH.
BE5F CD07BF                 CALL      COPY1     ;STATUS->MODUL
BE62 3A9FB9                 LD        A,(MGE) ;GER.EBENE HOLEN
BE65 CD1EBF                 CALL      MVOR
BE68 CD10BF                 CALL      COPY2     ;STATUS->RAM
BE6B 3A9FB9                 LD        A,(MGE) ;AKTUAL.EBENE
BE6E 329EB9                 LD        (MAE),A
BE71 CD20BF                 CALL      MVR1
BE74 CD4CBF                 CALL      RAME      ;SYST.RAM EIN
BE77 2168BF                 LD        HL,EXH
BE7A C330BF                 JP        IRAM      ;SYST.ZURUECK
BE7D              ;
BE7D              ;
BE7D              ;------ AB HIER NUR ANZEIGEN EBENE -----
BE7D              ;
BE7D              ;
BE7D F1           EBEN1     POP       AF        ;STACK OK MACHEN
BE7E 3A9EB9                 LD        A,(MAE)
BE81 CB2F                   SRA       A
BE83 CB2F                   SRA       A         ;BITS POSITIONR.
BE85 CD57BF                 CALL      AHEX
BE88 3E0D                   LD        A,0DH     ;NEW LINE
BE8A CD5CBF                 CALL      CRT
BE8D 3E0A                   LD        A,0AH
BE8F CD5CBF                 CALL      CRT
BE92 C368BF                 JP        EXH
BE95              ;
BE95              ;---------------------------------------
BE95              ;         SYs & RSYs
BE95              ;---------------------------------------
BE95              ;
BE95 7F7F                   DEFW      7F7FH
BE97 535973                 DEFM      'SYs'
BE9A 01                     DEFB      1
BE9B F3                     DI
BE9C 2168BF                 LD        HL,EXH
BE9F C329BF                 JP        IIRM
BEA2 7F7F                   DEFW      7F7FH
BEA4 52535973               DEFM      'RSYs'
BEA8 01                     DEFB      1
BEA9 F3                     DI
BEAA 2168BF                 LD        HL,EXH
```

kann mit dem Kommando EBENE bereits zwischen zwei im Speicher befindlichen Programmen auf unterschiedlichen Ebenen gewechselt werden, ohne daß eines der beiden Programme zerstört wird. Das hat z. B. beim Test des Debugging erhebliche Vorteile. Weiterhin wird es somit auch möglich, zwei oder mehrere völlig verschiedene System-Programme gleichzeitig in der Maschine zu haben und zwischen ihnen über das Kommando EBENE zu wählen. Beispielsweise kann in der Ebene 0 der BASIC-Interpreter aktiviert sein, mit einem Programm, in dem einige Maschinenroutinen benötigt werden. Die Ebene 1 kann dann mit dem Assembler aktiviert werden, um die entsprechenden Routinen zu erstellen. Danach kann wieder zum BASIC-Interpreter zurückgeschaltet werden, um die Routinen entsprechend einzubinden. Bei Weglassen des Parameters n wird die aktuelle Ebene angezeigt.

Funktion INIt

Sie wird unmittelbar nach dem Laden aufgerufen. Die Selbststartadresse ist BE00H, wo ein Sprung zu dieser Routine erfolgt (Selbststartadresse = Anfangsadresse). Es wird dann zunächst der entsprechende Modulsteckplatz „gesucht". (Modulsteckplatz 0B oder 0C). Wird das Strukturbyte F6H gefunden, so arbeitet das Programm weiter, andernfalls erfolgt eine Fehlermeldung, und das Programm kehrt in die CAOS-Eingabeschleife zurück.

Bei Vorhandensein des Moduls wird dieser automatisch eingeschaltet und der interne RAM nach Aufruf der Routine SYs abgeschaltet. Anschließend wird der gesamte RAM 0 bis 7FFFH gelöscht. Danach wird der nächste Block zugeschaltet und wieder gelöscht, bis alle 265 K mit 0 belegt wurden. Vor dem Rücksprung werden der Modul aus- und der interne RAM eingeschaltet. Letztlich erhalten die Systemzellen die ursprüngliche Belegung (130 bis 200 H – Routine RSYs).

4c

```
BEAD C330BF           JP      IRAM
BEB0          ;
BEB0          ;--------------------------------------
BEB0          ;++++   UNTERPROGRAMME   +++++++++++++++
BEB0          ;--------------------------------------
BEB0          ;
BEB0          ;       PARTST
BEB0          ;
BEB0          ;       TESTET  OB PARAMETER ZULAESSIG
BEB0          ;       UND POSITIONIERT RICHTIG IM BYTE
BEB0          ;
BEB0 F5       PARTST  PUSH    AF
BEB1 47               LD      B,A      ;PARAMETER->B
BEB2 3E03             LD      A,3      ;ZUL. MAXIMUM
BEB4 98               SBC     B
BEB5 DA61BF           JP      C,ERRX   ;WENN FEHLER
BEB8 F1               POP     AF
BEB9 CB27             SLA     A        ;BIT 1->3
BEBB CB27             SLA     A        ;BIT 0->2
BEBD F603             OR      3
BEBF C9               RET
BEC0          ;
BEC0          ;       TST     SUCHT MODUL
BEC0          ;               UND LEGT STECKPLATZADR.
BEC0          ;               IN (MSCH) AB
BEC0          ;
BEC0 2E08     TST     LD      L,8      ;STECKPLATZ
BEC2 CDD4BE           CALL    TST1
BEC5 2808             JR      Z,TSTEX
BEC7 2E0C             LD      L,0CH    ;STECKPLATZ
BEC9 CDD4BE           CALL    TST1
BECC C263BF           JP      NZ,ERROR
BECF 7D       TSTEX   LD      A,L
BED0 329DB9           LD      (MSCH),A;MERKER SCHACHT
BED3 C9               RET
BED4 3E01     TST1    LD      A,1
BED6 CD52BF           CALL    MOD1
BED9 7C               LD      A,H
BEDA FEF6             CP      0F6H     ;STRUKTURBYTE
BEDC C9               RET
BEDD          ;
BEDD          ;       MCRS
BEDD          ;       MCR
BEDD          ;       CLEAR
BEDD          ;
BEDD CDF0BE   MCRS    CALL    MCR
BEE0 CBD2             SET     2,D      ;X7 (2.EBENE)
BEE2 CDF0BE           CALL    MCR
BEE5 CBDA             SET     3,D      ;XF (4.EBENE)
BEE7 CDF0BE           CALL    MCR
BEEA CB92             RES     2,D      ;XB (3.EBENE)
BEEC CDF0BE           CALL    MCR
BEEF C9               RET
BEF0          ;
BEF0 CD50BF   MCR     CALL    MODU
BEF3 D9               EXX
BEF4 CDF9BE           CALL    CLEAR
BEF7 D9               EXX
BEF8 C9               RET
BEF9          ;
BEF9 210000   CLEAR   LD      HL,0     ;ANFANG
BEFC 3600             LD      (HL),0   ;"FUELLBYTE"
BEFE 110100           LD      DE,1     ;ANFANG + 1
BF01 01FE7F           LD      BC,7FFEH;LAENGE - 1
BF04 EDB0             LDIR
BF06 C9               RET
```

4d

```
BF07          ;
BF07          ;
BF07          ;       COPY1   kopiert 0000-3FFFH nach
BF07          ;                       4000-7FFFH
BF07          ;       COPY2   kopiert 4000-7FFFH nach
BF07          ;                       0000-3FFFH
BF07          ;
BF07 D9       COPY1   EXX
BF08 210000           LD      HL,0     ;QUELLE
BF0B 110040           LD      DE,4000H;ZIEL
BF0E 1807             JR      COP0
BF10 D9       COPY2   EXX
BF11 210040           LD      HL,4000H;QUELLE
BF14 110000           LD      DE,0     ;ZIEL
BF17 010040   COP0    LD      BC,4000H;COUNTER
BF1A EDB0             LDIR
BF1C D9               EXX              ;STATUS ZURUECK
BF1D C9               RET
BF1E          ;
BF1E          ;
BF1E          ;       MVOR    bereitet KDO. MODU vor
BF1E          ;
BF1E          ;
BF1E CBF7     MVOR    SET     6,A
BF20 57       MVR1    LD      D,A
BF21 3A9DB9           LD      A,(MSCH)
BF24 6F               LD      L,A
BF25 CD50BF           CALL    MODU
BF28 C9               RET
BF29          ;
BF29          ;
BF29          ;       IIRM    LEGT SYSTEM IN DEN IRM
BF29          ;       IRAM    LEGT SYSTEM NACH 130H-
BF29          ;               0200H (NORMAL)
BF29          ;
BF29 31D4BF   IIRM    LD      SP,0BFD4H
BF2C 3EBF             LD      A,0BFH
BF2E 1805             JR      IREX
BF30 31D401   IRAM    LD      SP,1D4H
BF33 3E01             LD      A,1
BF35 ED73AEB7 IREX    LD      (STCK),SP
BF39 1E31             LD      E,31H    ;SIXD
BF3B CD09F0           CALL    PV3
BF3E E9               JP      (HL)
BF3F          ;
BF3F          ;
BF3F          ;       MODAU   SCHALTET MODUL AUS
BF3F          ;       RAMA    SCHALTET SYST. RAM AUS
BF3F          ;       RAME    SCHALTET SYST. RAM EIN
BF3F          ;       MODU    UP NR.26H
BF3F          ;
BF3F          ;
BF3F 3A9DB9   MODAU   LD      A,(MSCH)
BF42 6F               LD      L,A
BF43 1600             LD      D,0
BF45 1809             JR      MODU
BF47 AF       RAMA    XOR     A
BF48 57               LD      D,A
BF49 6F               LD      L,A
BF4A 1804             JR      MODU
BF4C 1603     RAME    LD      D,3
BF4E 2E00             LD      L,0
BF50 3E02     MODU    LD      A,2
BF52 CD03F0   MOD1    CALL    PV1
BF55 26               DEFB    26H      ;MODU
BF56 C9               RET
```

```
BF57                ;
BF57                ;
BF57                ;        AHEX
BF57                ;        CRT
BF57                ;
BF57                ;
BF57 CD03F0   AHEX   CALL    PV1
BF5A 1C              DEFB    1CH      ;AHEX
BF5B C9              RET
BF5C CD03F0   CRT    CALL    PV1
BF5F 00              DEFB    0        ;CRT
BF60 C9              RET
BF61                ;
BF61                ;
BF61                ;        ERRX
BF61                ;        ERROR
BF61                ;        EXH
BF61                ;
BF61                ;
BF61 F1       ERRX   POP     AF       ;STACK SAUBER !
BF62 F1              POP     AF
BF63 1E19     ERROR  LD      E,19H
BF65 CD09F0          CALL    PV3
BF68 1E12     EXH    LD      E,12H    ;LOOP
BF6A FB              EI               ;INT EIN !
BF6B CD09F0          CALL    PV3
BF6E C300E0          JP      REST
BF71                ;
BF71                ;++++++++++++++++ ENDE ++++++++++++++++++
BF71                ;

ERRORS: 0000
```

Bilder 4a-e Programmausdruck

Funktion SYs

Durch Aufruf werden die Systemzellen des Betriebssystems CAOS in den IRM verlagert. Es ist danach z. B. das Abschalten des System-RAM mit dem Befehl
SWITCH O O(Enter)
möglich. Es sei an dieser Stelle nochmals auf die Besonderheiten des IRM, wie höhere Zugriffszeit und sichtbare Speicherzugriffe, verwiesen.

Funktion RSYs

Das Kommando ist ebenfalls ohne Parameter einzugeben, und es bewirkt die Umkehrung des Kommandos SYs. Es werden also die Systemzellen wieder in den Adreßraum von 0130H bis 0200H rückverlegt. Dabei ist selbstverständlich zu beachten, daß in diesem Adreßbereich ein RAM aktiviert sein muß, der nicht schreibgeschützt sein darf. Wird das nicht beachtet, sind Abstürze unvermeidlich.

Weitere Hinweise

Durch Nutzung insbesondere der letzten zwei Menüfunktionen wird der Anwender in die Lage versetzt, auch mit dem Modul M011 die Effektivität der Arbeit zu verbessern, weil es damit ermöglicht wird, z. B. ein BASIC-Programm und eine Assemblerquelle gleichzeitig zu bearbeiten, indem die Adressierung entsprechend geschickt gewählt wird.

Der Modul als großer Datenspeicher

Wird der interne RAM als Programmspeicher genutzt, so kann der 256-K-RAM-Modul als großer Datenspeicher eingesetzt werden. Durch die Verschiebung der Blöcke im 64-K-Adreßbereich kann praktisch jeder 16-K-Block des 256-K-RAM in den Adreßbereich 4000H bis 7FFFH „geschoben“ werden. Dazu kann vom Anwender im BASIC und in FORTH die Anweisung SWITCH und auf Maschinen- bzw. Assemblerniveau das CAOS-Unterprogramm MODU (UP.Nr.: 26H) aufgerufen werden. Die Daten lassen sich damit sehr schnell nach bestimmten Merkmalen durchsuchen und/oder ändern. Die Zugriffszeit bei einer solchen Arbeitsweise des RAM beträgt gegenüber einer Diskette nur 1/4 bis 1/100.

Ausblick

Natürlich bleibt es dem Anwender überlassen, weitere Funktionen zu implementieren. Denkbar wäre z. B. die Organisation des gesamten Speichers in 16-K-Blöcken. Ein weiterer interessanter Einsatzfall dürfte die Verwendung des Moduls als elektronische Diskette sein.

Literatur

/1/ Modul-M011-Beschreibung. VEB Mikroelektronik „Wilhelm Pieck“ Mühlhausen

/2/ Modul-M005-Beschreibung. VEB Mikroelektronik „Wilhelm Pieck“ Mühlhausen

/3/ KC85/3-System-Handbuch. VEB Mikroelektronik „Wilhelm Pieck“ Mühlhausen

Termine

Fachtagung FORTH

WER? Interessengemeinschaft FORTH des KdT-Bezirksverbandes Suhl
WANN? 20. bis 22. April 1988
WO? Suhl
WAS? System- und Anwendersoftware der Programmiersprache FORTH
WIE?
Anmeldungen nimmt der KdT-Bezirksverband, Straße der Opfer des Faschismus 29, PSF 510, Suhl, 6000 entgegen. Vortragsangebote bitte mit kurzer Inhaltsangabe an die Interessengemeinschaft FORTH, Dr. Finsterbusch, PSF 190, Ilmenau, 6300.

Fehlerhaft

Bedauerlicherweise sind die im Artikel „Integrierter Systemtaktgenerator DL 8127 D“ enthaltenen Bilder 5, 6 und 8 (siehe MP 7/1987, S. 213) fehlerhaft. Wir veröffentlichen deshalb nachstehend die korrigierten Bilder.

Autor und Redaktion

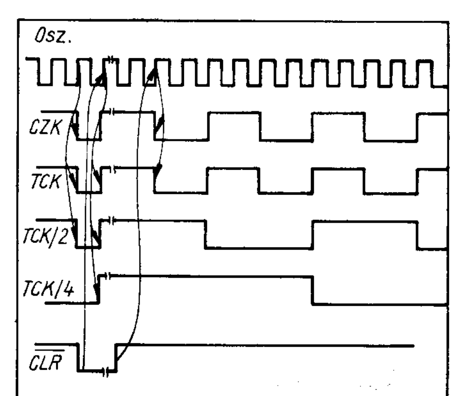


Bild 5 Taktdiagramm U-8000-Modus

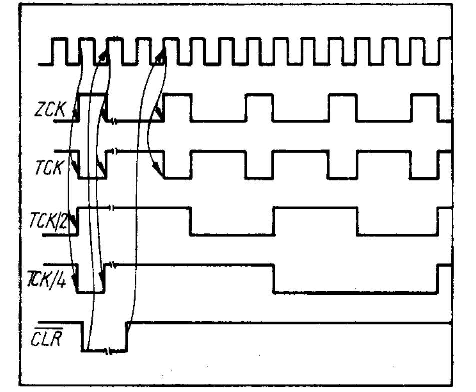


Bild 6 Taktdiagramm 8086-Modus

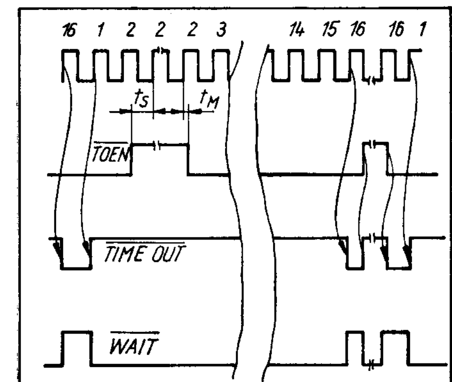
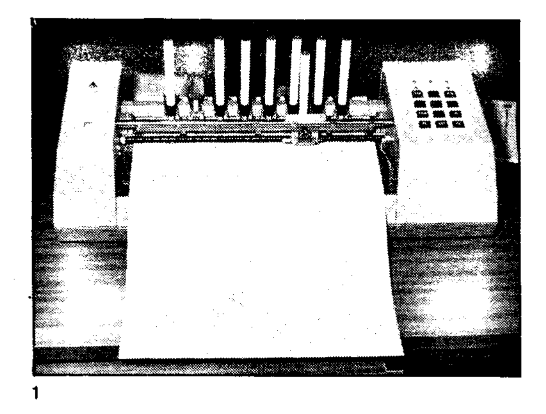
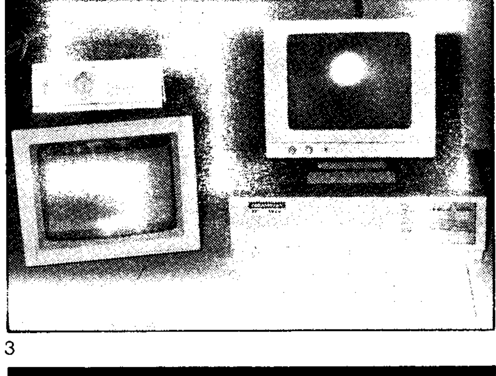
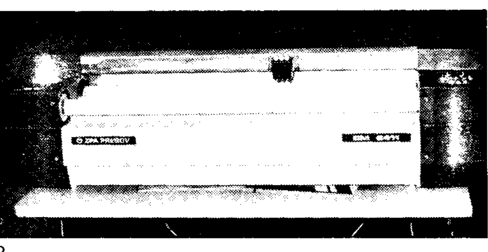


Bild 8 TIMEOUT-Zeitverhalten

In diesem Jahr öffnete die 29. Internationale Maschinenbaumesse Brno ihre Tore. Der Vorjubiläumsjahrgang dieser wichtigsten Messe unseres Nachbarlandes ČSSR, sie fand vom 16. bis 23. September statt, stand unter dem Motto „Progressive Technologien im Maschinenbau". Zur gleichen Thematik fand parallel zum Messegeschehen die „Konferenz der Redakteure der technischen Presse" mit großer Beteiligung ausländischer Journalisten statt.
Wegen der Rekonstruktion stand weniger Ausstellungsfläche zur Verfügung, dennoch waren etwa 2900 Aussteller – 160 mehr als 1986 – zu verzeichnen.
Für den langjährigen Messebesucher sofort sichtbar traf das auch auf die Computertechnikanbieter zu. Denn ihr traditionelles Domizil, die Halle D, reichte nicht mehr. So mußten einige Firmen mit verkleinerten Quadratmeterzahlen Ausstellungsfläche vorliebnehmen und andere in kleine Ausstellungspavillons auf Wege und Flächen ausweichen. Auch die Halle C, in der Meßtechnik, Kommunikationstechnik und – für Applikationen in diesem Sinne – Computertechnik zu sehen war, bot keine Reserven mehr. Was für das Messeunternehmen ein großer Erfolg ist, nämlich viele Aussteller, ist für den Fachbesucher ein „Marathon". Jahr für Jahr mehr Firmen und jedes Unternehmen bietet eine größere Typenvielfalt. Unter dem Blickwinkel sogenannter internationaler Industriestandards nimmt die scheinbare Vielfalt doch wieder einheitliche Formen an.
Nachfolgend wollen wir ausgewählte Exponate vorstellen und Trends aufzeigen.


1


3


2

29. Internationale Maschinenbaumesse Brno

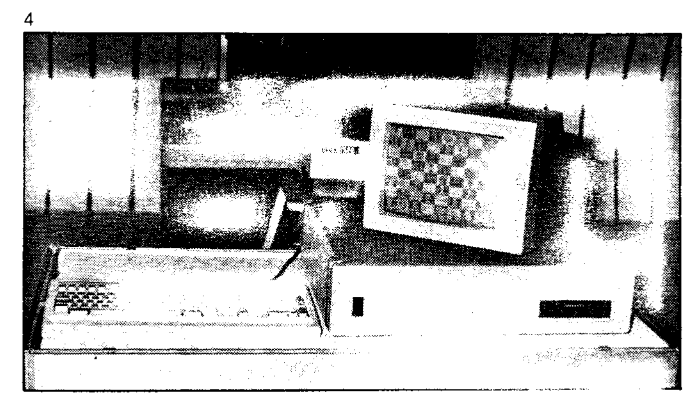
Die ČSSR spezialisierte sich im Rahmen des RGW u. a. auf grafische Ausgabetechnik und hat dadurch auf diesem Gebiet große Erfahrungen. In der DDR sind z. B. viele Zeichenautomaten DIGIGRAF im Einsatz. Der auf dem Messestand von KOVO vorgestellte DIGIGRAF unterschied sich äußerlich kaum von seinem Vorgänger, neu an ihm war die verbesserte Elektronik. Zur Besucherattraktion entwickelte sich der Grafikplotter **COLORGRAF** – die vollständige Bezeichnung des Herstellers ARITMA Prag lautet: Grafische Ausgabeeinrichtung ARITMA 0512–COLORGRAF (Bild 1). Über den COLORGRAF 0512 können Zeichnungen mit bis zu acht auswechselbaren Schreibfedern erstellt werden. Dabei ist es möglich, die Zeichnungen, in Abhängigkeit von den eingesetzten Schreibfedern, entweder mehrfarbig oder mit unterschiedlicher Linienstärke anzufertigen. Die Aufzeichnung in x-Richtung wird mittels Schreibfederbewegung ausgeführt, die in y-Richtung durch Bewegen des Papiers. Die Grundschrittweite (Raster) beträgt 0,125 mm. Der eingebaute μP sorgt für einigen Komfort. So sind im internen Programmspeicher des Plotters Routinen zum Zeichnen verschiedener Arten von Linien (volle, unterbrochene, strichpunktierte), Kreisen, Kreisbögen sowie ein kompletter Satz ASC II – und diakritischer Zeichen enthalten. Die Software des COLORGRAF ist kompatibel zu der vergleichbarer Plotter von Hewlett-Packard. Nun noch kurz einige technische Daten:

- Nennbreite des verwendeten Arbeitsmaterials 297 mm
- Zeichenbreite 264 mm, das entspricht 2112 Schritten
- Genauigkeit ± 0,05 mm
- maximale Zeichengeschwindigkeit 80 mm/s
- Schnittstelle CCITT V 24/V 28
- Senkungs- und Stabilisierungsdauer der Schreibfedern 50 ms
- Anhebedauer der Schreibfedern 50 ms
- Leistungsaufnahme 62 W
- Abmessungen 530 × 205 × 165 mm³
- Masse 6,2 kg.

Während der Messe dürften die ausgestellten Plotter durchaus einen Dauertest absolviert haben. Von morgens bis abends wurden für Besucher „Demonstrationszeichnungen" angefertigt (siehe 4. Umschlagseite), wobei der COLORGRAF für eine derartige Zeichnung durchschnittlich 20 Minuten benötigte.

Zeichnungen auf einem Arbeitsfeld von 860 × 24 000 mm² kann der **VZ 930** (CM 6411) (Bild 2) erstellen. Er verfügt über eine maximale Zeichengeschwindigkeit von 100 mm/s. Die Grundschrittweite beträgt 0,05 mm. Plott- und Wiederholgenauigkeit sind mit ± 0,15 mm angegeben. Der VZ 930 besitzt 3 Stifte, mit denen er auf perforiertes Papier zeichnet. Bei einer Größe von 1550 × 410 × 630 mm³ hat das Gerät eine Masse von 65 kg. 250 W Leistung werden aufgenommen. An Schnittstellen sind V24 und IFFS vorhanden.

An Computersystemen offerierte KOVO Weiterentwicklungen bzw. spezielle Konfigurationen bereits bekannter Hardware. Beispielsweise die interaktive grafische Station IGS 2 auf Basis des M 16–22 von ZVT Banska Bystrica, an der auch der VZ 930 angeschlossen war.

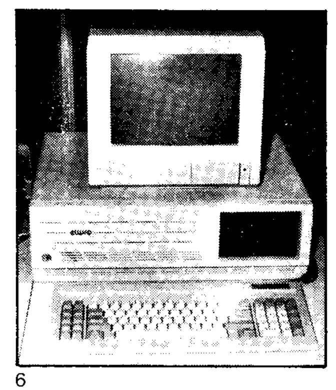
4


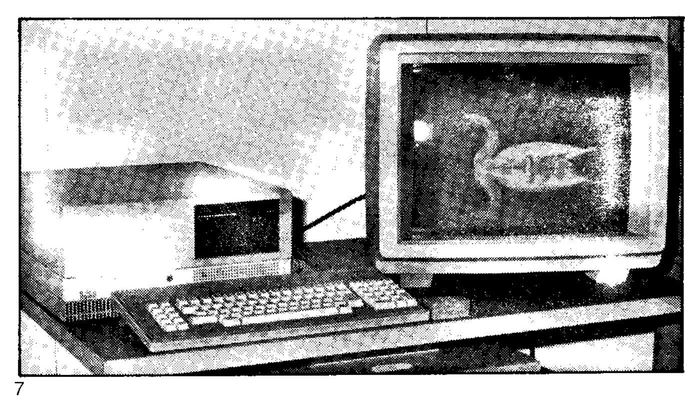
5


Die ČSSR steht vor dem Problem, daß es zwar verhältnismäßig viele Entwicklungen unterschiedlicher Computertypen gibt, sie aus den verschiedensten Gründen aber kaum in größerer Stückzahl gefertigt werden.
Um so mehr Beachtung verdient die Zielstellung, bis 1990 die Produktion schnell zu entwickeln und 150 000 bis 200 000 Mikrorechner der Volkswirtschaft zur Verfügung zu stellen. Einen großen Anteil wird dabei die 8-Bit-Technik ausmachen.

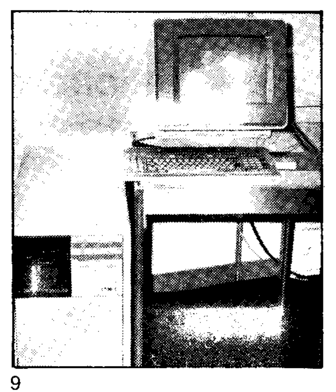
Eines war fast allen Ausstellern gemeinsam – das Angebot IBM-PC-kompatibler Computer. In der ČSSR ist es der PP 06, allerdings keine Messeneuheit, deswegen soll an dieser Stelle nicht ausführlicher auf ihn eingegangen werden. Nachfolgend stellen wir einige Entwicklungen kurz vor. Über den **EC 1834** von Robotron (Bild 3) mit µP K1810 WM 86 haben wir bereits in MP 11/87, 2. US kurz informiert. Aus der VR Bulgarien kommt der **EC 1832** (Bild 4) mit einem Prozessor 8088. Der Koprozessor 8087 kann als Option eingesetzt werden. Weitere Daten sind: 640 KByte Hauptspeicher, Hard-Disk mit 5 oder 10 MByte, 1 Floppy-Laufwerk mit 360 KByte. In größeren Stückzahlen soll in Bulgarien der **Prawez 16** (Bild 5) zum Einsatz kommen, ebenfalls mit µP 8088 und Koprozessor 8087, außerdem 2 Floppy-Laufwerken, oder eine Floppy-Einheit und eine Hard-Disk, 256 KByte RAM, Farb- oder Monochrombildschirm.

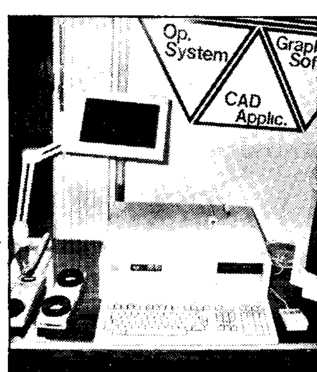
Das rumänische Außenhandelsunternehmen Electronum zeigte u. a. den **FELIX PC** mit Mikroprozessor 8086, 2 Floppy-Laufwerken, 640 KByte RAM und Farbmonitor.
Die polnische Außenhandelsfirma ELWRO stellte einen AT-kompatiblen PC vor, den ELWRO 801 AT (Bild 6): Mikroprozessor i 80286, Floppy-Disk mit 1,2 MByte, Hard-Disk mit 20 MByte, Betriebssystem MS-DOS.
Der **Trident AT** wird vom jugoslawischen Unternehmen Iskra Delta vertrieben. Neben 5¼-Zoll-Floppy-Disk mit 1,2 MByte verfügt der Computer über eine 80-MByte-Winchester-Disk und eine 40-MByte-Streamer-Kassette. Als Betriebssystem kommt MS-DOS 3.2 zum Einsatz. Bekannte internationale Firmen wie Hewlett-Packard, IBM, ICL, NCR, Tektronix, Wang, Rocc,


6

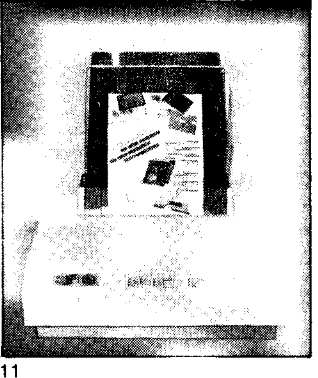

7


8

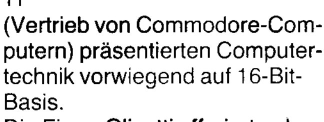

9



10


11

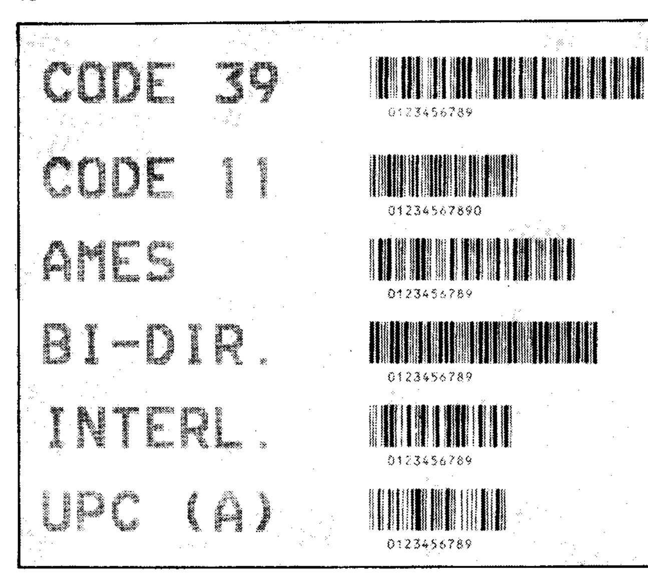
12



(Vertrieb von Commodore-Computern) präsentierten Computertechnik vorwiegend auf 16-Bit-Basis.
Die Firma Olivetti offerierte als Neuheit das System **PE 28** (Personal Engineering) (Bild 7) mit der CPU 80 286 (8 MHz), schneller Arithmetikeinheit 80 287, 1 MByte RAM, 1,2-MByte-Floppy-Einheit, 20-MByte-Hard-Disk und Interface (RS-232C/V24 und Centronics).
Der 19-Zoll-Grafikbildschirm erlaubt die gleichzeitige Darstellung von 256 Farben aus einer möglichen Palette von 4096 bei einer Auflösung von 1280 × 960 Punkten.
Als Betriebssysteme werden MS-DOS und XENIX verwendet.
Dicht umlagert waren natürlich die Modelle des PS/2 von IBM (siehe auch MP 8/87, S. 243). Das Modell 30 war im RGW bereits auf den Messen in Poznan, VR Polen, und Budapest, UVR, zu sehen. Das leistungsstärkste Computersystem dieser Familie „Modell 80" soll ebenso wie das Betriebssystem OS/2 erst ab 1988 zur Auslieferung kommen. Unser Foto zeigt das Modell 60 (Bild 8).

32-Bit-Technik stellte die französische Firma Bull mit ihrem **SPS 7** (Bild 9) vor. Der SPS 7/70/75 basiert auf Mikroprozessor

13



68 020 (16,67 MHz) und Koprozessor 68 881. Als Hauptspeichergröße werden 4 MByte angegeben, aufrüstbar bis maximal 16 MByte. Das SPS 7 verfügt über Winchester-Disk und Streamer-Tape. Bis zu 4 grafische Bildschirme (monochrom oder color) sind an ein System anschließbar.
Neben den großen Firmen war das Bild auf der Internationalen Maschinenbaumesse Brno aber auch geprägt durch viele kleinere Handelsfirmen, hauptsächlich aus Österreich, der BRD und Schweiz, die Hard- und Software von Unternehmern aus vielen Ländern offerierten, wobei die Produzenten häufig in Taiwan, Südkorea oder Hongkong ansässig sind.
Die kleine österreichische Handelsfirma array-Data GmbH bot für IBM-kompatible PCs einen extrem flachen, verstellbaren Bildschirm an, den sogenannten **FLAT-SCREEN** (Bild 10). Die Abmessungen sind 330×230×25 mm³, die Bildfläche beträgt 250×160 mm². 25 Zeilen mit je 80 Zeichen sind bei 8×8 Punkten je Zeichen darstellbar, oder im Grafikmodus beträgt die Auflösung 640×200 Punkte.

Der **GAMMACOLOR VIDEO-PRINTER** (Bild 11) enthält einen schnellen Videodigitalisierer mit Bildspeicherung. Die maximale Abspeicherzeit bei 1280 (vertikal) × 1024 (horizontal) Bildpunkten und 4096 Farben beträgt 0,3 s. Die Auflösung ist mit 4,8 Punkten pro mm angegeben. Die Bilder werden aus den vier Grundfarben Gelb, Magenta, Cyan und Schwarz zusammengesetzt. Je nach Format braucht der Videoprinter 2 bis 4 Minuten für die Ausgabe eines Bildes, ohne dabei den Bildschirm zu blockieren.
Ein sogenannter **TOPAS Powermaker** soll den PC vor Spannungsspitzen, Spannungsabfällen und -ausfall schützen, indem er weiter den ordnungsgemäßen Betrieb gewährleistet. Dabei gibt es eine Vielzahl von Modellen in Abhängigkeit von benötigter Leistung, Spannung und Frequenz.
60 dB leise ist der neue Drucker **VT 23 900** (Bild 12) von VIDEOTON. Bei einem Satz von 96 Zeichen schafft er 762 Zeichen je Minute. Dreimal ist jedes Zeichen auf dem Metallband angeordnet, das mit hoher Geschwindigkeit rotiert. Für jede Druckposition existiert ein „Hammer". Jeder 9. Hammer kann gleichzeitig schlagen. An Schnittstellen stehen V 24 und Centronics zur Verfügung.
Als ersten Matrix-Zeilendrucker der Welt bezeichnet MANNESMANN Tally seinen **MT 690**/**MT 660** mit einer Druckleistung von 900 bzw. 600 Zeilen je Minute (Schnelldruck). In Bild 13 ist ein Druckmuster abgebildet.
Der MT 690 druckt mit 132 Druckelementen, dabei hat eine Zeichenmatrix je nach geforderter Qualität eine Rastergröße von minimal 7×9 (Höhe und Breite) und maximal 15×25 Zeichen (Schönschriftqualität). Für Schönschrift beträgt die Druckgeschwindigkeit 450 Zeilen je Minute. 850×980×572 mm³ sind die Abmessungen und als Gewicht werden etwa 160 kg genannt.
Die DDR war auf der Internationalen Maschinenbaumesse größter sozialistischer Aussteller. Der Außenhandelsbetrieb Robotron stellte, neben dem bereits erwähnten EC 1834, Arbeitsplatzcomputer A 7150 (siehe im gleichen Heft), PC 1715 und Schreibmaschinen aus. Der PC 1715 wurde 1987 und wird 1988 in großen Stückzahlen in die ČSSR exportiert.

I. P.

Fotos: Paszkowsky (12)

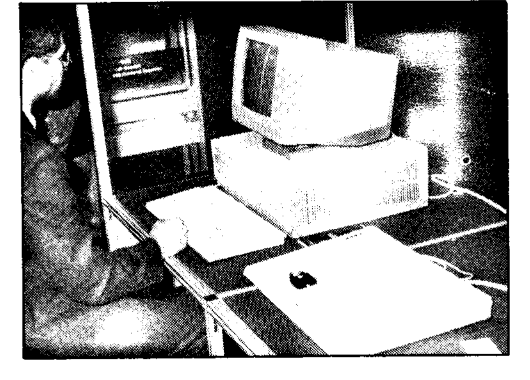
Vorgestellt

Unter dieser neuen MP-Rubrik werden Computer und Peripheriegeräte in ihren wichtigsten Parametern vorgestellt – ausführlicher als in der Rubrik „Technik international".



In dieser Ausgabe beginnen wir mit dem

Arbeitsplatzcomputer A 7150



Der *Arbeitsplatzcomputer robotron A 7150* ist modular aufgebaut und besteht aus dem Rechnergrundgerät, dem Bildschirm und der Tastatur. Basis der zentralen Verarbeitungseinheit ist ein 16-Bit-Mikroprozessor. Die Rechnerarchitektur, der international standardisierte Systembus, die hohe Speicherkapazität und die Grafikfähigkeit ermöglichen eine nutzerfreundliche Anwendung.

Die Grundausstattung umfaßt:
- Rechnergrundgerät
10 Steckplätze für Logikmoduln, 2×5,25"-Diskettenspeicher in Slimline-Ausführung mit je 1 MByte Speicherkapazität und vorbereitetem 1×5,25"-Festplattenspeicher mit 30 bis 50 MByte Speicherkapazität
- Bildschirmeinheit
Einfarbig/mehrfarbig, dreh- und schwenkbar, 31-cm-Bildröhre, geeignet für Bildschirmsteuerungen entweder durch eine alphanumerisch-quasigrafische, 2000 Zeichen im Raster 80×25, oder eine rastergrafische Ansteuerung mit 640×480 Bildpunkten
- Tastatur
48 Zeichentasten, Realisierung von 2 verschiedenen Zeichensätzen entsprechend nationalen Standards – lat./dt. oder lat./kyrill. – durch Mehrfachbelegung der Tasten, numerische Tastatur, Funktionstastatur, Anzeige unterschiedlicher Arbeitsmodi

Als periphere Geräte können wahlweise genutzt werden:
- Drucker robotron K 6313, K 6314, Baureihe K 6320, 1152/257
- Plotter robotron K 6418 und K 6411
- Digitalisiergeräte robotron K 6401 und K 6404
- Grafisches Tablett robotron K 6405

Der robotron A 7150 ist konstruktiv und logisch modular aufgebaut. Die Logikmoduln arbeiten in einem international standardisierten Mikrorechnerbus (I-41). Auf Grund der Verwendung eines universellen Bussystems und der durchgängigen Gestaltung gehört der robotron A 7150 zur Gruppe der offenen Systeme.

Software

Das Hardwarekonzept wird durch Systemsoftware ergänzt, die kompatibel zu international weitverbreiteten Betriebssystemen ist. Das Grundsoftwarepaket enthält folgende Komponenten
- Betriebssysteme
- Compiler
- Standardsoftware

Betriebssysteme
Für den kommerziellen Einsatz:
- Betriebssystem DCP 3.1 (kompatibel MS-DOS3.1)
- Betriebssystem SCP 1700

Für Echtzeiteinsatzfälle und Multiprogrammbetrieb:
- Betriebssystem BOS 1810

Für Time-sharing-Betrieb:
- Betriebssystem MUTOS 1700

Compiler
- BASIC (unter SCP 1700-Interpreter – unter DCP 3.1-ANSI Standard-BASIC)
- FORTRAN 77 (unter SCP 1700, DCP 3.1 und BOS 1810)
- PL/M (unter BOS 1810)
- COBOL (unter SCP 1700 und DCP 3.1)
- MODULA 2 (unter SCP 1700 und DCP 3.1)
- PASCAL (unter SCP 1700 und DCP 3.1)
- Sprache C (unter SCP 1700 und DCP 3.1)

Standardsoftware
- AIDOS/M 16 Informationsrecherchesystem
- DIALOG/M 16 Maskengenerator
- LIST/M 16 Listenprozessor
- TABCALC/M 16 Tabellenrechnung
- TEXT 40/M 16 Textverarbeitungssystem
- REDABAS/M 16 Datenbanksystem
- TESYS-3/M 16 Technologisches System zur Softwareentwicklung
- TEPROS-SCP 1700 Textverarbeitungssystem, kommandoorientiert
- GRAFIK/M 16 Geschäftsgrafik
- NUMATH/M 16 Numerische Mathematik

Technische Daten
- Zentrale Verarbeitungseinheit

Mikroprozessor
K 1810 WM 86 (5 MHz, 16 Bit)
Befehlsausführungszeit
0,4 μs (Register-Register)

Speicheradreßraum	1 MByte
E/A-Adreßraum	64 KByte
Interruptsystem	9 Ebenen
Numerikprozessor	K 1810 WM 87
• RAM	256 ... 768 KByte
Zugriffszeit	560 ns

- Externspeicher

Kapazität 2 MByte; 2×5,25"-Diskettenspeicher robotron K 5601
30–50 MByte, 1×5,25"-Festplattenspeicher
- Interface

IFSS, IFSP; IFSP-M (Centronics), S2 (V.24)
- Energieversorgung

220 V +10% / −15%
47 ... 63 Hz
- Funkentstörung nach VDE 0871
- Maße (B×T×H), mm

Rechnergrundgerät	170×486×451
Tastatur	50×461×239
Bildschirmeinheit	323×338×354

CAD/CAM-Schlüsseltechnologie als Intensivierungsfaktor

Autorenkollektiv unter Leitung von Prof. Dr. D. Kochan. Verlag Die Wirtschaft Berlin 1987. 320 S., Sachwortverzeichnis, 20,– M

Man braucht nicht zu zögern, um zu erklären, daß die vorliegende Veröffentlichung nicht nur Fachleute der Praxis anspricht, sondern auch mit geeignet ist, als Nachschlagewerk und Lehrhilfsmittel zu dienen. Zu dieser Aussage berechtigt eine eindeutige Abfolge der Stoffvermittlung in Verbindung mit praxisorientierten und methodologisch fixierten Aussagen. Dominant sind im Kapitel 1 und 2 Aussagen zu CAD/CAM-Systemkomponenten und -Systemaufbau, Schnittstellen, internationale Standards und Modelle, rechnergestützte Angebots- und Auftragsbearbeitung sowie Erzeugnisentwurf und technologische Fertigungsvorbereitung, -durchführung und automatisierte Fertigung. Die materiell-technische und software-technologische Basis (Kapitel 3) beschäftigt sich u. a. mit Hauptbestandteilen von CAD/CAM-Anlagen, ihren Funktionen, ihrer Struktur und Architektur sowie peripheren Geräten.

Beachtenswert sind eindeutige Aussagen zur Software-Technologie sowie zu erforderlichen Datenbasen (Datenbanken) und für CAD/CAM zur Verfügung stehende Datenbankbetriebssysteme. Auch Vergleiche zwischen unterschiedlichen Computersystemen werden auf der Grundlage ausgewählter Parameter angeboten. Unterstützt werden diese Darstellungen in einem Anhang, in dem eine Übersicht über derzeitig verfügbare CAD/CAM-Rechentechniken und periphere Geräte angeboten wird.

Ausgewählte Beispiellösungen und Einsatzerfahrungen im Kapitel 4 sind besonders informativ, da u. a. auch Auskünfte zur eingesetzten Gerätetechnik, Software und eingetretenen Nutzen vermittelt werden. Ökonomische, organisatorische und arbeitswissenschaftliche Aufgaben insbesondere unter den Aspekten Vorbereitung von CAD/CAM-Lösungen, Ermittlungen ihrer Effektivität sowie arbeitswissenschaftliche Erfordernisse bei der CAD/CAM-Einführung enthält Kapitel 5. Insgesamt ein Buch, das nur empfohlen werden kann, auch Leiter unterschiedlicher Ebenen sind angesprochen, die im Hinblick auf Planung und Realisierung von CAD/CAM-Systemen Entscheidungen vorzubereiten und zu treffen haben.

Prof. Dr. W. Schoppan

Wissensspeicher BASIC

von P. Heblik, Volk und Wissen Volkseigener Verlag Berlin 1986, 1. Auflage, 302 S.

Publikationen zur Programmiersprache BASIC sind in der DDR bereits einige erschienen. Dabei ist BASIC-Buch nicht gleich BASIC-Buch. So gibt es in der methodischen Aufbereitung des Stoffes häufig Unterschiede. Speziell für den Leserkreis „Einsteiger" scheint der *Wissensspeicher BASIC* zugeschnitten zu sein. Zu Recht als Wissensspeicher bezeichnet, ist die vorliegende Publikation aber auch ein geeignetes Lehrbuch zur Einführung in die Sprache BASIC. Ausführliche Erläuterungen und viele Programmbeispiele erleichtern das Verständnis des dargebotenen Stoffes.

I. Paszkowsky

Computer Graphics Programming GKS – Graphics Standards

von Enderle, G.; Kansy, K.; Pfaff, G.
Springer-Verlag Berlin, Heidelberg–New York–Tokyo, 542 S.

Das wohl wichtigste Anwendungsgebiet der Computergrafik sind heute rechnergestützte Entwurfsverfahren. Sie werden unter dem Begriff „Computer Aided Design" (CAD) zusammengefaßt und stehen im Blickpunkt einer breiten Nutzergemeinschaft in den unterschiedlichsten Bereichen der Volkswirtschaft.
Im Rahmen verschiedener internationaler Standardisierungsgremien wurden seit 1975 umfangreiche Anstrengungen unternommen, um die wesentlichen Softwareinterfacebedingungen auf diesem Gebiet einer Normung zu unterziehen. An dieser Tätigkeit waren auch Mitarbeiter aus der DDR beteiligt. Das Ergebnis dieser Arbeit ist der GKS-Standard.
Von drei ehemaligen Mitgliedern der ISO-Arbeitsgruppe zur GKS-Standardisierung wurde nun ein Buch vorgelegt, das die Programmierung von Computergrafik unter Nutzung des „Graphical Kernel Systems" (GKS) beschreibt. Es handelt sich dabei um eine wohl einzigartig umfassende Darstellung des angesprochenen Problemkreises, die in vier Kapitel untergliedert wurde:

– Eine ausführliche Einführung in die Konzepte der Computergrafik unter Berücksichtigung der GKS-Prinzipien.
– Einen Überblick über die wissenschaftlich-technische Entwicklung, die zur Herausbildung des GKS-Standards geführt hat.
– Eine umfangreiche Erläuterung der einzelnen GKS-Funktionen und ihre Anwendung in einer von den verschiedenen Programmiersprachen unabhängigen Notation.
– Eine Behandlung der verschiedenen Aspekte einer konkreten GKS-Implementation, wie Betriebssystemfragen, Hardwareabhängigkeiten, Besonderheiten im Zusammenhang mit unterschiedlichen Programmiersprachen u. a. m.

Dem Rezensenten etwas verwunderlich ist der geradezu spartanische Umgang mit erläuterndem Bildmaterial in einem Buch über Computergrafik, das sonst auch vom herausgebenden Verlag in einer vorbildlichen Ausstattung vorgelegt wurde.

Dr. L. Claßen

Expertensysteme

Von P. Schnupp und U. Leibrandt. Springer-Verlag Berlin–Heidelberg–New York–Tokyo 1986, Compass-Reihe, 140 S.

Expertensysteme ist ein Fachbuch, aber nicht nur für Fachleute. Es wendet sich in erster Linie an jene, die sich mit den Problemkreisen künstliche Intelligenz und Expertensysteme unter dem Blickwinkel des künftigen Anwenders auseinandersetzen wollen bzw. müssen.
Anschaulich und leicht verständlich werden in 10 Kapiteln Fragen beantwortet wie:
Was ist „künstliche Intelligenz"?
Warum Expertensysteme?
Wann und wie?
Wie funktioniert eine „Interferenzmaschine"?
Wie programmiert man „wissensbasiert"?
Natürliche Sprache als Computer-Eingabe?
Als positiv wurden weiterhin die Hinweise zu weiterführender Literatur am Ende eines jeden Kapitels empfunden. Im Anhang befinden sich ein kleines Lexikon der Schlagwörter, das 67 Begriffe kurz erläutert, und ein PROLOG-Programmbeispiel.

I. Paszkowsky

Graphisches Kernsystem (GKS)

Beuth Verlag Berlin 1986

Das Graphische Kernsystem (GKS) ist eine rechnerunabhängige Zusammenstellung von Standardfunktionen für unterschiedlichste Aufgabenstellungen bei der Programmierung von grafischen Darstellungen. Die durch den Beuth Verlag/West-Berlin jetzt vorgelegte funktionale Beschreibung der GKS-Norm (DIN 66252) sieht die Hauptgründe für die Einführung des GKS-Grundsystems in den Zielstellungen:
– eine leichte Übertragbarkeit (Portierbarkeit) von Anwendungsprogrammen auf unterschiedliche Rechner sicherzustellen
– den Anwendungsprogrammierer beim Verstehen und Verwenden graphischer Methoden maximal zu unterstützen und
– die funktionelle Standardisierung von grafischen Ein-/Ausgabegeräten (Farbsichtgeräte, Plotter ...) voranzutreiben.

Das Buch gibt einen ausgezeichneten Überblick über die Grundlagen, Funktionen und Datenstrukturen der GKS-Norm. Eine wichtige Begriffserläuterung ist ebenso enthalten, wie die ausführliche Beschreibung der einzelnen GKS-Funktionen. Die Zuordnung der Funktionen zu verschiedenen GKS-Leistungsstufen (0a, 0b, 0c, 1a, 1b, 1c, 2a) trägt einem breiten Anwenderspektrum, von der Zeichnung statischer Bilder bis hin zur Darstellung dynamischer Vorgänge unter Echtzeitbedingungen, Rechnung.

Dr. L. Claßen

Netzwerkanalyse mit Mason-Graphen

Von Dr.-Ing. Ulrich Mende. VEB Verlag Technik, Reihe Informationselektronik. 156 Seiten, 64 Bilder, 8 Tafeln, Broschur, DDR 16,50 M, Ausland 24,– DM. Auslieferung durch den Fachbuchhandel. Bestellangaben: 553 734 6/Mende, Netzwerkanalyse.

Der Autor bietet vor allem Entwicklungsingenieuren und Studenten der Elektrotechnik/Elektronik eine topologische Methode zur Berechnung elektrischer und elektronischer Netzwerke. Diese Methode beruht auf der Anwendung des Mason-Graphen und kann auch vorteilhaft in Verbindung mit Mikrorechnern genutzt werden.
Die Darlegungen sind bewußt ausführlich gehalten, so daß sich die Broschüre auch sehr gut zum Selbststudium eignet. Zahlreiche praktische Beispiele fördern das Verständnis.

dBase-Listengenerator

Für die Arbeit mit dBASE-Kommandodateien wurde ein Serviceprogramm erstellt, welches folgende Funktionen realisiert: – Syntaxkontrolle auf ordnungsgemäßes Schließen aller DO WHILE-Schleifen sowie aller IF- und DO CASE-Anweisungen
– Formatierung des Quelltextes
– Laufzeitoptimierung
– Listengenerator für Einzelblattverarbeitung (mit Angabe von Zeilennummern, Doppeldruck aller DO-WHILE- und IF-Anweisungen, auszugsweisem Druck von Programmteilen, Kommentaren hinter ENDIF und ENDDO u. a.)
Gegenüber anderen dBASE-Hilfsprogrammen bietet es den Vorteil, daß es ohne Verlassen des Datenbanksystems als Kommandodatei gestartet werden kann. Es ist vollständig menügesteuert. Bei Bedarf können on-line Hilfsinformationen eingeblendet werden. Das Programm läuft unter den Betriebssystemen SCPX und CP/A und ist für dBASE und Epson LX-86 entwickelt. Auf Wunsch kann eine Version für REDABAS bzw. andere Druckertypen abgegeben werden.
VEB Kyffhäuserhütte Artern, Rudolf-Breitscheid-Str. 15/16, Artern, 4730, Abt. ODV; Tel. 7492 (Koll. Trautvetter)
Niebur

Druckeransteuerung für mehrere Computer

Zur Verbesserung der Auslastung der Drucktechnik wurde auf Basis der V.24-Schnittstelle eine Ansteuerung entwickelt, die den Anschluß mehrerer Computer mit V.24-Interface an einem Drucker gestattet. Dabei ist kein Umstecken bzw. Umschalten zwischen den Computern erforderlich. Die Verdrahtung kann fest erfolgen, was zur Erhöhung der Lebensdauer und zur Verbesserung des Arbeitsschutzes beiträgt. Die bisher verwendete Software kann unverändert weitergenutzt werden. Die maximale Leitungslänge beträgt etwa 15 Meter.
Die Schaltung wird seit längerer Zeit mit drei KC 85/3, PC 1715 und MRES benutzt.
Die Unterlagen können nachgenutzt werden.
VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, Eisenacher Str. 40, Mühlhausen, 5700.
Schiwon/Kirves

Rechnungsprogramm

Das Rechnungsprogramm für PGH und Handwerksbetriebe der Elektrobranche nach PAO 564 wurde in BASIC auf einem Schneider Joyce unter CP/M 3.0 geschrieben und arbeitet mit 2 Laufwerken (auch auf PC 1715 möglich). Im A-Laufwerk sind Programm, Kundenadresse sowie Rechnungsdaten abgelegt. Im B-Laufwerk sind sämtliche Preise gespeichert.
Mit dem Programm wird insbesondere die Rechnungsführung mit den Koeffizientenumrechnungen erleichtert.
Elektro-Handwerk Joachim Brustmann, Hauptstraße 18, Schönwalde, 1291; Tel. Mühlenbeck 601 *Brustmann*

Programmpakete „Geodätische Software"

Zur Nachnutzung angeboten werden die Programme
- einfache Punktbestimmungen
- Höhenbestimmungen
- freie Standpunktwahl/Schnitte
- Transformationen
- mathematische Berechnungen
- Krümmen
- sonstige Berechnungen
– Kranbahnkontrolle
– räumlicher Vorwärtseinschnitt
– Druckerprogramm-Koordinatenverzeichnis aus FSI-Daten für BC A 5120/5130, PC 1715 im BS SCP.
VEB Kombinat Geodäsie und Kartographie, Forschungszentrum, Leitstelle EDV, Gohliser Str. 4, Leipzig, 7022, Tel. 56340 *Cramer*

Drucker-Driver für ATARI 1029

Zum Anschluß des Druckers ATARI 1029 an Mikrorechner auf der Prozessorbasis U880 wurde ein Driver entwickelt. Als weitere Hardware-Voraussetzung wird lediglich ein PIO-Port benötigt. Der Driver wandelt den byteseriellen Datenstrom des Wirtssystems in die dem Drucker verständlichen Frames um und sendet diese bitseriell. Der Driver liegt in an die Betriebssysteme der Kleincomputer KC 85/1, 85/2, 85/3 und 87 anpassungsfähiger Form vor, ist aber leicht an die Betriebssysteme der Büro- und Personalcomputer oben genannter Prozessorbasis anzupassen.
Technische Universität Dresden, Sektion Informationstechnik, Bereich Kommunikations- und Computertechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4635121 *Herzmann*

Magnetbandtreiber KASS

Der angebotene MB-Treiber arbeitet unter dem Betriebssystem UDOS. Er ermöglicht ein schnelles Kopieren von Dateien auf Kassette. Damit ist er besonders zur Archivierung von Quellprogrammen, Texten und Binärdateien geeignet.
VEB Numerik „Karl-Marx", TN, PSF 947, Karl-Marx-Stadt, 9010.

CMOS-Prüfgerät

Das Prüfgerät ermöglicht die statische und dynamische Funktionsprüfung und Fehlersuche an CMOS-Schaltkreisen in der Schaltung einschließlich Erkennung von Einzelimpulsen. Es weist geringe Abmessungen, jedoch keine Prüfstiftform auf. Die Kontaktgabe erfolgt durch Anklemmen der Meßleitung an dem zu untersuchenden Pin des Schaltkreises. Die Speisung des Prüfgerätes geschieht mittels der Betriebsspannung der getesteten Baugruppe (5 bis 15 V), die Anzeige der Auswerteergebnisse durch eine 7-Segment-Anzeige.
VEB MLW Medizintechnik Leipzig/Forschungszentrum Dresden, Overbeckstr. 48, Dresden, 8030.

TURBO-PASCAL Nutzerhandbuch

Es wurde von uns ein Nutzerhandbuch TURBO-PASCAL erarbeitet, das auch die Anwendung von PASCAL 880/S unterstützen kann (Autor Dipl.-Phys. W. Finze).
Die 2. Auflage berücksichtigt auch die Anwendung von TURBO-PASCAL auf 16-Bit-Systemen mit Betriebssystemen entsprechend CP/M-86 und MS-DOS.
Für Anwender IBM-kompatibler Technik liegt zusätzlich eine Ergänzung vom Nutzerhandbuch vor, in der die besonderen Möglichkeiten von TURBO-PASCAL für diese Technik beschrieben werden.
Interessierte Betriebe, die das Handbuch in größeren Stückzahlen benötigen und die es im Kleinoffsetverfahren selbst vervielfältigen wollen, können über einen Nachnutzungsvertrag die Druckvorlagen und Belegexemplare erhalten.
Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Direktorat R, WB Informationsverarbeitung, Richard-Wagner-Straße 31, Warnemünde, 2530.
Prof. Dr. Mennenga

Arbeitskräftestruktur

Die Softwarelösung für den PC 1715 dient zur Rationalisierung der Bearbeitung des Stellenplanes in den Kombinaten und Betrieben.
Das Programm gliedert sich in
1. Datenpflegeprogramm
2. Analysenprogramm Arbeitskräftestruktur und
3. Analysenprogramm Lohnfonds.
Die Programmteile 2 und 3 ermöglichen, die innere Verflechtung unter verschiedenen Gesichtspunkten (z. B. Beschäftigtengruppe, Gehaltsgruppe, Schichtstimulierung und Schichtrhythmus und Lohnfondsinanspruchnahme) zu analysieren.
VEB Forschung, Entwicklung und Rationalisierung des Schwermaschinen- und Anlagenbaus, Bleckenburgstr. 25, Magdeburg, 3011

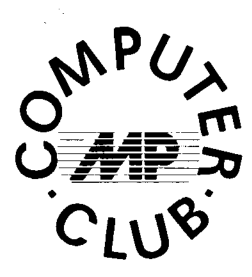
Abrechnung und Recherchen zum Produktionsplan

Mit dem Programmpaket können neben der Abrechnung der Produktion über eine Menüführung des Bedieners unterschiedliche Recherchen angestellt werden. Genutzt werden PC 1715 mit in REDABAS erstellten Programmen, die erweiterungsfähig sind. Das Produktionssortiment umfaßt etwa 1200 Geräte, es werden 4 Dateien aktualisiert. Benötigt werden 2 Disketten mit einer Kapazität von je 630 KByte.
VEB Funkwerk Kölleda, Eugen-Richter-Straße, PSF 48, Kölleda, 5234

Wir suchen ...

... zur Nachnutzung eine Interface-Multiplex-Baugruppe, die es gestattet, mindestens 3×V.24-, 1×IFSS-, 1×SI1.2-Schnittstellen an eine V.24-Schnittstelle anzuschließen.
Kombinat VEB Elektronische Bauelemente, Ingenieur- und Projektierungsbüro, Köpenicker Allee 114–118, Berlin-Waldesruh, 1147; Tel. 6440111 (Abt. RLR, Koll. Kirsch) *Marschall*

... ein Zusatzprogramm für BC A 5120 unter SCPX, mit dem auch bei Verwendung von Standardsoftware (TP, REDABAS) die alternative Nutzung der rechten und/oder linken Bahn des Druckers SD 1152 möglich ist.
Deutsche Reichsbahn, Ibr-Fa, Außenstelle Leipzig, Werkstättenstraße 8, Engelsdorf, 7123.
Hoffmann



A4-Plotter für robotron-KC

Dr. Gert Keller
VEB Robotron-Meßelektronik
„Otto Schön“ Dresden

Für die Kleincomputer KC 85/1 und KC 87 vom VEB Robotron-Meßelektronik „Otto Schön“ Dresden werden jetzt Kleinplotter des Prager Kooperationsbetriebes Laboratorni pristroje angeboten. Die wichtigsten Leistungsmerkmale dieses bereits zur LFM'87 vorgestellten Computer-Plotter-Systems sollen im folgenden skizziert werden.

Leistungsmerkmale

Der Kleinplotter XY 4131 zeichnet auf einer Fläche von 255 × 180 mm^2 mit einer Zeichen- bzw. Adressiergenauigkeit von 0,1 mm. Er arbeitet nach dem Rollenplotterprinzip mit einer Geschwindigkeit von etwa 8 cm/s. Wegen der flexiblen Stifthalterung ist es möglich, verschiedene handelsübliche Stifte (Faserstifte, Kugelschreiber, Bleistifte, Tuschestifte) einzusetzen. Der Anschluß des Plotters erfolgt an die E/A-Buchse des Kleincomputers (bzw. an einen E/A-Erweiterungsmodul). Zusätzlich sind noch ein Plotter-Grafik-Modul mit der erforderlichen BASIC-Spracherweiterung und ein von Kassette zu ladendes Treiberprogramm GRPLOT erforderlich. Alle drei Komponenten (Kleinplotter, Plotter-Grafik-Modul und Treiberkassette) werden nur gemeinsam durch den Vertriebsbetrieb (VEB Robotron-Vertrieb Berlin, Mohrenstr. 62, Berlin, 1086) angeboten.
Die Software ist im Speicher der 2. RAM-Erweiterung ab Adresse 9800H angeordnet, so daß für den Nutzer etwa 38 KByte für BASIC-Programme verfügbar bleiben.

BASIC-Spracherweiterung

Mit dem Plotter-Grafik-Modul erhält der Nutzer eine auf die Bedürfnisse der Kleinrationalisierung abgestimmte BASIC-Spracherweiterung, die (für diese Computerklasse) relativ leistungsfähige Anweisungen zur Ausgabe grafischer Darstellungen bereitstellt.
Diese Anweisungen lassen sich etwa den folgenden vier Gruppen zuordnen:

Plotterzuweisung und Initialisierung

Die Zuweisung des Plotters erfolgt durch eine SCREEN-Anweisung, die das Bereitstellen und Einbinden eines Grafikmoduls berücksichtigt. Dem Rücksetzen des Plotterstiftes bzw. des Koordinatensystemes dienen die Anweisungen
HOME
Stift auf aktuellen Nullpunkt setzen
GCLS
Einstellen Standard-Koordinatensystem und Stift auf Nullpunkt setzen

Zeichenanweisungen

Die Zeichenanweisungen
PSET
Punkt setzen
LINE
Linien und Rechtecke zeichnen
CIRCLE
Kreise, Kreisbögen und Ellipsen zeichnen
wurden in Syntax und Wirkung dem internationalen Standard (z. B. BASICA des IBM PC/XT) angeglichen und dienen der Ausgabe der üblichen grafischen Grundelemente.

Koordinatensysteme

Durch die speziellen Anweisungen
ZERO
Festlegung des Nullpunktes (auch außerhalb der Zeichenfläche)
SCALE
Festlegung von Maßstabsfaktoren für die x- und y-Achse (auch Achsenspiegelung) kann der Nutzer ein eigenes (Welt-) Koordinatensystem definieren und damit aufwendige Skalierungsrechnungen für seine Darstellungen vermeiden.

Textausgabe

Für die Ausgabe von Texten auf den Plotter stehen die Anweisungen
LABEL
Ausgabe einer Zeichenkette ab aktueller Stiftposition
SIZE
Festlegung von Breite und Höhe der Buchstaben sowie Schreibrichtung und Schräglage zur Verfügung. Mit ihnen lassen sich sehr flexible Beschriftungen von grafischen Darstellungen ausführen.

Einsatzmöglichkeiten

Auf Grund seiner Leistungsmerkmale ist der Kleinplotter XY 4131 besonders für Ausbildungs- und Qualifizierungsaufgaben und für kleinere Rationalisierungsaufgaben geeignet. Es können sowohl mathematische Funktionen als auch kleinere technische Zeichnungen, Standardformulare oder auch grafische Protokolle von Meßvorgängen dargestellt werden. Als sehr nützlich erweist sich bei allen Anwendungen die freie Wählbarkeit eines problemangepaßten Koordinatensystems.
Die Bilder zeigen Einsatzmöglichkeiten des beschriebenen Computer-Plotter-Systems.

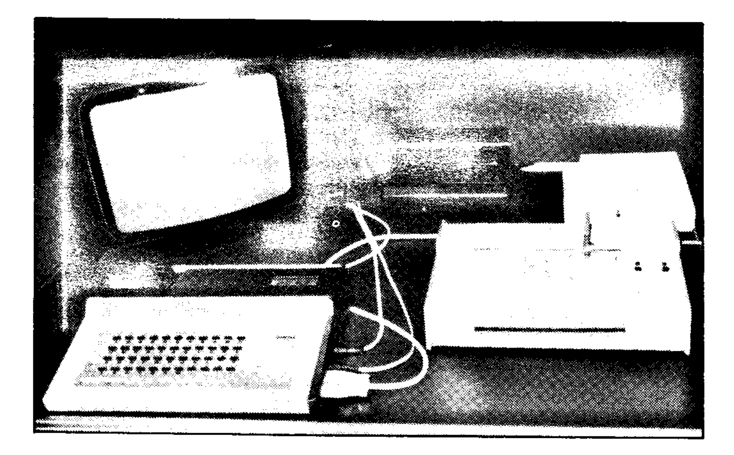
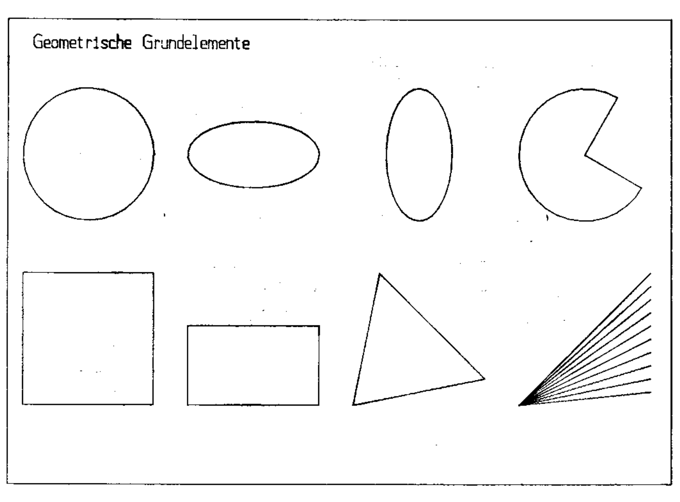
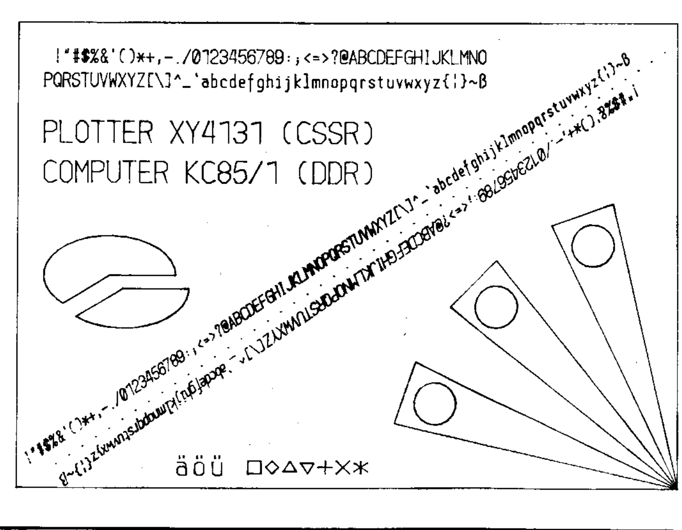


Foto: Weiß





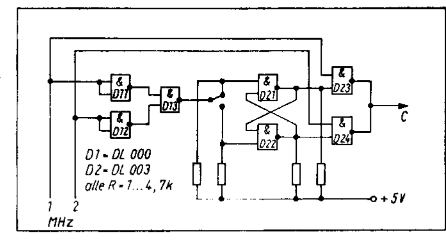


Bild 1 Variante 4

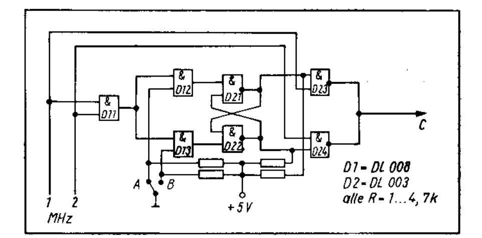


Bild 2 Variante 5

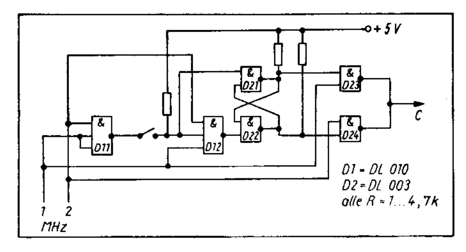


Bild 3 Variante 6

Taktfrequenzumschaltung MRB Z1013

(Teil 2)

Gegenwärtig existieren zwei Varianten des MRB Z1013. Während ältere MRB-Platinen noch mit Anfallbauelementen bestückt wurden und mit einer Taktfrequenz von 1 MHz betrieben werden, sind auch MRB-Platinen im Vertrieb, die mit einer Taktfrequenz von 2 MHz arbeiten. Daraus resultiert das Problem, daß Software von Kassetten nicht ohne Schwierigkeiten auf beide Varianten übernommen werden kann. Es ist zwar möglich, mit einem Zwei-Geschwindigkeits-Tonbandgerät die Programme ein- und auszulesen, aber auch dieser Weg ist nicht problemlos.

Aus diesem Grunde wurde nach elektronischen Lösungen gesucht. Einfache Schaltungen haben den Nachteil, daß es durch Kontaktprellen des Umschalters zu einem unsauberen Taktsignal kommen kann. Obwohl in der Praxis recht selten, kann es dabei zum Programmabsturz oder zu Bitveränderungen im RAM kommen.

Mit dem Einsatz von zwei zusätzlichen IS lassen sich weitere Varianten elektronischer Taktfrequenzumschaltungen realisieren.

Die Bilder 1 bis 3 zeigen mögliche Schaltungsvarianten.

Allen gemeinsam ist das aus D2.3 und D2.4 gebildete Tor. Die Ansteuerung des Tors erfolgt über ein aus D2.1 und D2.2 gebildetes RS-Flip-Flop. Dies sorgt für ein sauberes Umschaltsignal und die Entprellung des Schalters. Die Umschaltung des Taktes erfolgt, ohne daß der Prozessor davon etwas „merkt“, d. h., aufgrund der Verknüpfung der beiden Taktsignale durch D1 kann eine Umschaltung nur erfolgen, wenn beide Teilerausgänge zur Taktversorgung auf gleichem Potential liegen. Vorteilhaft an Variante 3 ist, daß diese nur einen einfachen Schalter benötigt. Variante 2 dagegen läßt sich einfach softwaremäßig steuern. Dazu ist es notwendig, an den Punkten A und B statt des Schalters einen Ausgabeport (PIO oder Latch) anzuschließen.

Abschließend muß noch darauf hingewiesen werden, daß bei Umbauten am MRB Z1013 die Garantie des Herstellers VEB Robotron-Elektronik Riesa erlischt.

Computer-Club robotron
A. Köhler

☒ KONTAKT ☎

VEB Robotron-Anlagenbau, Computer-Club, PSF 180, Leipzig, 7010

Erfahrungsaustausch gefragt

Am 18. März 1987 nahm unser Computerclub mit 109 Mitgliedern die Arbeit auf. Der Computerclub besteht aus 12 Arbeitsgruppen, wobei in jeder Arbeitsgruppe 8 bis 10 Studenten einer Fachrichtung arbeiten. Es werden Themen bearbeitet, die vor allem für den Schulbetrieb genutzt werden können. So sind es z. B.:

– Programme, die es ermöglichen, den Unterricht rechnergestützt durchzuführen
– Programme zur schnellsten Lösung mathematischer Routineberechnungen bei studentischen Laborübungen
– Programme zur Dokumentation.

Zur Lösung gerätetechnischer Aufgabenstellungen wurde eine Arbeitsgruppe Hardware gebildet.

Dem Computerclub stehen Kleincomputer KC 85/1 und KC 85/3 zur Verfügung. Um bei so vielen Mitgliedern einen reibungslosen Betrieb zu gewährleisten, erfolgt die Vergabe von Rechenzeiten nach Vorbestellung. Um auch an den Wochenenden eine Arbeit am Rechner zu ermöglichen, wurde im Studentenwohnheim ein Raum eingerichtet, (ausgerüstet mit zwei KC 85/1 und zwei KC 85/3) der dem Computerclub zur Verfügung steht.

Wir würden gern mit anderen Computerclubs in Erfahrungsaustausch treten, um Tips und Anregungen zu bekommen.

Kukura

☒ KONTAKT ☎

Ingenieurschule „Ernst Thälmann“, Computerclub, Senftenberg, 7840

FABAS

Lange frei definierbare Festkomma-Arithmetik in BASIC

Prof. Dr. Horst Völz, Berlin

Für viele Anwendungen, u. a. bei ökonomischen Rechnungen, reicht die Genauigkeit des BASIC-Interpreters nicht aus. Soll auf 1 Pfennig genau gerechnet werden, so bieten 6 gültige Stellen als größten Wert 9999,99 M. Genau hier stellt u. a. das zu beschreibende Programmsystem neue Möglichkeiten bereit. Es besteht aus einem Maschinenteil, der fest in BASIC eingebunden ist, und der für die Grundrechenarten bis zu 35 gültige Stellen, davon einen festdefinierbaren Teil hinter dem Komma, zuläßt. Den BASIC-Teil für die Grundroutine zum Erproben zeigt Bild 1. Den Hexdump der ersten, sehr langen BASIC-Zeile 0 (nicht im BASIC-Programm gelistet, da das zu Fehlern führen würde) zeigt Bild 2. Es wurde bewußt der Screen-Dump gewählt, damit beim Eingeben schnell Fehler erkennbar sind. Das Programm läuft auf dem KC 85/2 (3). Es gibt eine etwas abweichende Variante auch für den KC 85/1 bzw. KC 87.

Eingabe des Programms

Es wird zunächst der Hexcode mit MODIFY eingegeben. Dann wird BASIC aufgerufen, und es werden die Zeilen des BASIC-Programms eingegeben. Das Programm muß unmittelbar (noch vor jedem RUN) gerettet werden. Beim RUN entstehen nämlich in der Zeile 0-Werte. Werden dennoch LIST oder EDIT oder SAVE usw. aufgerufen, wird die Zeile 0 und damit das wichtige Maschinenprogramm zerstört. Für die Entwicklungsarbeit existieren aber zwei Hilfsroutinen, die alle Probleme beheben lassen:

Mit CALL*434 werden alle 0-Werte des Maschinenprogramms in FF verwandelt. Dann ist BASIC editierbar, ladbar usw. Um wieder ein arbeitsfähiges Programm für RUN zu haben, werden mit CALL*429 alle FF des Maschinenprogramms in 0 zurückgewandelt: Hier lag ein Problem; das Maschinenprogramm mußte so geschrieben werden, daß weder 0 noch FF im Befehlscode vorkommt.

Routinen des Maschinen- und BASIC-Programms

Im gemeinsamen Maschinen- und BASIC-Programm existieren folgende Funktionen:

a) Eingabe von Zahlen (E)
b) Anzeige von Zahlen (A)
c) Fehlertestroutine (F)
d) Registerübergabe (T) Inhalt von N nach M
e) Arithmetik: +, −, *, /, % (REG1 Operator REG2, Ergebnis nach REG∅)
f) Rundungsroutine
g) Hilfsroutine für Programmierarbeiten

Die Funktionen a) bis c) werden über GOSUB, die Funktionen d) bis g) werden über CALL * aufgerufen.

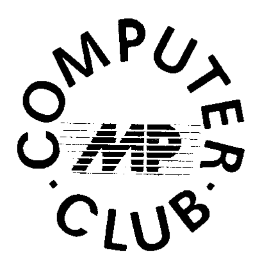
Die Programme des Maschinencodes sind:

CALL*40A

Rundungsmaßroutine für eine Zahl. Die Zahl wird um 1 erhöht bei der letzten Ziffer (am Ende Zahl, also meist Nachkommazahl); vorher Register N festlegen.
DOKE 1056, FN R(N)

CALL*429

wandelt Maschinencode in arbeitsfähige Version um



FF → 0
Es ist dann nicht mehr möglich:
Zeilen
· löschen
· editieren
· eingeben
· speichern usw.

CALL*434
wandelt Maschinencode in speicherbare, editierbare (einschließlich RENUMBER) Form um
0 → FF

CALL*453
verlagert den Inhalt von Register N nach Register M, vorher
DOKE 1056, FN R(N)
DOKE 1058, FN R(M)

CALL*469
Addition: (REG0) = (REG1) + (REG2)

CALL*4D2
Subtraktion: (REG0) = (REG1) – (REG2)

CALL*534
Multiplikation: (REG0) = (REG1) · (REG2)

CALL*5CF
Division: (REG0) = (REG1)/(REG2)

CALL*5D2
Prozent: (REG0) = (REG1)/(REG2) * 100

Bei allen Arithmetikroutinen erfolgt eine exakte Rundung. Ergibt sich bei der Berechnung ein Überlauf, so kann eine Fehleranzeige mittels BASIC-Zeile 210 erfolgen. Beim Fehler ist

(RS) = 0

Bei Fehleranzeige ist nach Betätigung irgendeiner Taste der gültige, niederwertige Zahlenteil mit richtigem Vorzeichen im REG 0 abgelegt.

Definition von Registern

Die Register 0, 1 und 2 dienen der Arithmetik. Sie werden ab Adresse 0 angelegt. Ihre Größe hängt von der definierten Stellenzahl ab. Weitere Register sind möglich und können an beliebigen freien Speicherplätzen angelegt werden. Es möge Y die Zahl der gültigen Stellen (geradzahlig) sein und X die Nachkommastellen (ebenfalls geradzahlig) mit

$2 < X < Y;$

dann werden beim Initialisieren des Programms zwei spezielle Speicherplätze in der langen Zeile gemäß

SB = Z/2 (Stellenbyte)
NK = X/2 (Nachkommabyte)

belegt. Der Grund dafür ist, daß eine BCD-Arithmetik verwendet und 1 Byte für das Vorzeichen genutzt wird.
Hierfür gilt:

0 – 254 = „+"
255 = „–"

Die Registerlängen betragen folglich SB + 1. Bezeichnet werden die Arithmetikregister mit einer natürlichen Zahl N. So wird die Anfangs- und Endadresse gemäß der definierten Funktion in Zeile 230 berechnet:

FN R = N * (SB + 1) (Anfang)
FN R = (N + 1)*(SB + 1) – 1 (Ende)

Die Speicherplätze für die Arbeitszellen zur Funktion des Programms sind wie folgt festgelegt:

Dezimal	HEX	Bedeutung
1058/9	422/3	Adresse 0
1060/1	424/5	Adresse 1
1062	426	Stellenbyte SB
1063	427	Nachkommabyte NK
1064	428	Reservebyte RS

Ein- und Ausgaberoutinen

Die Ein- und Ausgaberoutinen wurden in BASIC geschrieben. Sie benötigen daher etwas mehr Zeit. Diese Zeit ist von der gewählten Stellenzahl abhängig. (Je größer die Stellenzahl, desto länger ist die Zeit für die Ein- und Ausgabe). Die Ausgabe aller Zahlen erfolgt generell rechtsbündig, ohne führende Nullen, aber mit dem Vorzeichen (nur negatives) ganz links. Die Eingabe akzeptiert vielfältige Formate, allerdings mit folgenden Voraussetzungen:

a) „–" oder „+" zu Beginn der Zahl („+" kann entfallen)



Bild 2 Hexcode des Maschinenprogramms

b) Die Zahl darf nur einen Dezimalpunkt enthalten (sonst fehlerhafte Eingabe ohne Fehleranzeige, der erste Punkt wird gewählt)
c) Das Komma ist in der Zahl nicht zulässig (Alle Ziffern nach dem Komma werden ignoriert.)

Bild 1 Grundroutine zum Erproben

```
1 !-------------------------*
2 !     *** FABAS ***       *
3 !Festkomma-Arithmetik BASIC *
4 ! H.Voelz          14.8.86  *
5 !-------------------------*
6 !GOTO230=Start
7 !Anzeigeroutine
8 !==============
10 L=FNR(N):X=SB-NK+1:IFPEEK(L)=255THENPRINT"-";:ELSEPRINT" ";
20 Y=0:FORI=1TOSB:A=PEEK(L+I):IFI=XTHENPRINT".";:Y=1
30 B=(AAND240)/16:A=AAND15
40 IFY+A+B=0THENPRINT"  ";:NEXT:RETURN
50 IFY+B=0THENPRINT" ";RIGHT$(STR$(A),1);:Y=1:NEXT:RETURN
60 Y=1:PRINTRIGHT$(STR$(B),1);RIGHT$(STR$(A),1);:NEXT:RETURN
65 !Eingaberoutine
66 !=============
70 L=LEN(A$):S$=LEFT$(A$,1):C=0:D=0:IFS$="-"THEND=255
80 IFS$="-"ORS$="+"THENL=L-1:A$=RIGHT$(A$,L)
90 A$="00"+A$:L=L+2:X=L+1
100 FORI=1TOL:IFMID$(A$,I,1)="."THENX=I:I=L
110 NEXT:Y=NK+NK-L+X
120 IFY<>0THENL=L+Y
130 IFY>0THENFORX=1TOY:A$=A$+"0":NEXT
140 IFY<0THENC=VAL(MID$(A$,L+1,1)):A$=LEFT$(A$,L)
150 Y=FNR(N)+SB:
160 FORX=L-1TOL-SB-SB-2STEP-2:A=0:IFX=L-NK-NK-1THENX=X-1
170 IFX 0THENA=VAL(MID$(A$,X,2))
180 B=INT(A/10):A=16*B+A-10*B:POKEY,A:Y=Y-1
190 NEXT:IFC=>5THENDOKE1058,FNR(N)+SB:CALL*40A
200 POKEFNR(N),D:RETURN
205 !Ueberlaufanzeige
206 !================
210 IFPEEK(1064)THENPRINT"Ueberlauf":INPUTX
220 RETURN
225 !Start main
226 !==========
230 DEFFNR(N)=N*(SB+1):DEFFNE(N)=(N+1)*(SB+1)-1
235 !ab hier Arbeisversion
236 !=====================
240 INPUT"Stellenzahl = ";SB:SB=INT(SB/2+.5):POKE1062,SB
250 INPUT"Nachkommazahl = ";NK:NK=INT(NK/2+.5):POKE1063,NK
260 PRINT:INPUT"1.Zahl = ";A$:IFA$="T"GOTO380
270 IFA$="E"GOTO390
280 IFA$="A"GOTO400
290 IFA$="F"GOTO310
300 N=1:GOSUB70:INPUT"2.Zahl = ";A$:N=2:GOSUB70
310 INPUT"Funktion";A$:IFA$="+"THENCALL*469:GOTO370
320 IFA$="-"THENCALL*4CF:GOTO370
330 IFA$="*"THENCALL*531:GOTO370
340 IFA$="/"THENCALL*5CC:GOTO370
350 IFA$="%"THENCALL*5CF:GOTO370
360 PRINT"Falsche Eingabe":GOTO260
370 GOSUB210:N=0:GOSUB10:GOTO260
380 INPUT"REG. N- M";N,M:DOKE1058,FNR(N):DOKE1060,FNR(M):CALL*453:GOTO260
390 INPUT"REC";N:INPUT"Zahl = ";A$:GOSUB70:GOTO260
400 INPUT"REG";N:GOSUB10:GOTO260
```

d) Die Zahl darf nicht mehr Stellen vor dem Dezimalpunkt enthalten, als das Maximalformat erlaubt (keine Fehleranzeige, Übernahme der möglichen niederwertigen Stellen)
e) Die Nachkommazahl kann beliebig lang oder kurz sein (Ergänzung durch zusätzliche Nullen bzw. exakte Rundung erfolgen automatisch)
f) Führende Nullen vor dem Dezimalpunkt können entfallen.
g) Bei ganzen Zahlen kann der Dezimalpunkt entfallen.

Es seien 10 gültige Stellen und 2 Nachkommastellen festgelegt;
Gültige Eingaben sind dann:
12
1234.5
1234.567 intern auf 1234.57 gerundet
−12.
−12.3
−.33
.45
Fehlerhafte Eingaben infolge Komma sind:
123,3 intern 123
1153,7 intern 1153
Fehlerhafte Eingaben sind weiter:
1.234.56 zwei Punkte; intern: 1.23
1A3A.57 enthält A; intern: 1030.57
Alle nicht gültigen Zeichen werden als 0 interpretiert. Zu lange Zahlen
1234567890.33
verlieren die vorderen Stellen – intern wird erhalten
45678901.33

Variablen und Hilfsprogramme

Globale Variablen sind:
N Registernummer; NK Nachkommabyte; SB Stellenbyte; RS Reservebyte
Lokale Variablen sind:
Eingabe
A enthält Zahl als ASCII-Kette; A, B, C, D, I, L, X, Y, S
Ausgabe
A, B, I, L, X, Y

Beschreibung des Hilfsprogrammes:
Mit dem Hilfsprogramm kann wie mit einem Taschenrechner gerechnet werden. Es ist bezüglich Geschwindigkeit nicht voll optimal gestaltet. Es erlaubt folgende Eingaben bei Anzeige:
1. Zahl =
T = Transport von Register N nach Register M
E = Eingabe einer Zahl in Register N
A = Anzeige der Zahl im Register N
F = Funktionswahl für REG1: REG2 → REG0 alle Zahleneingaben gehen ins Register 1. Bei Anzeige:
2. Zahl =
kann nur eine Zahl ins Register 2 eingegeben werden.
Bei Anzeige „Funktion" sind möglich:
„+", „−", „*", „/" und „%".
Mit diesen Befehlen sind vielfältige Rechnungen möglich.

Anwendung der Programme

Das vorgestellte Programm dient nur zum Erlernen des Umganges. Für die praktische Anwendung muß ein spezielles Programm geschrieben werden. Dabei muß aber der BASIC-Kern bis Zeile 230 (ohne REM-Zeilen) erhalten bleiben. Für die Registerbenennung werden die definierten Funktionen genutzt.

Magnetbandkatalog für KC 85/2,3

Es ist im allgemeinen sehr mühevoll, die auf den Magnetbändern gespeicherten Programme systematisch zu katalogisieren und die Programmdatei ständig auf dem aktuellen Stand zu halten. Neu hinzukommende Programme verändern das Bild sehr schnell. Es ist daher wünschenswert, eine automatische Dokumentation der Bänder vornehmen zu können. Das angefügte BASIC-Programm übernimmt diese Aufgabe:
Nach dem Start des Programms wird die zu katalogisierende Kassette in den Computer eingespielt. Es erfolgt dann ein selbständiges Lesen aller Kopfdaten. Ist das Band durchgelaufen, liegt eine vollständige Dokumentation auf dem Bildschirm bzw. auf dem Drucker (bei Ersetzen der PRINT-Befehle durch PRINT # 2) vor. Die Annehmlichkeiten dieses Programms wird man bald zu schätzen wissen. *Prof. Dr. H. Junek*

```
10 REM ****************
20 REM * H E A D E R  *         (C)  JUNEK 87
30 REM ****************
40 :
100 CLS
110 I=12288: REM #3000
120 READ X: IF X>=0 THEN POKE I,X:I=I+1: GOTO120
130 DATA 205,24,240,205,3,240,10,205,27,     240,201,-1
140 DEF FNP(X)=VPEEK(X)+256*VPEEK(X+1)
150 PRINT "H E A D E R          START  LGE    AUTO"
160 CALL*3000
170 IF PEEK(498)<>1 THEN160
180 B=14080:REM B=BUFFER-#8000
190 I=0
200 TYP$="CODE"
210 IF VPEEK(B)=211 THEN I=I+3:TYP$="PGM "
220 IF VPEEK(B)=212 THEN I=I+3:TYP$="DATA"
230 GOSUB300:IF I<11 THEN230
240 PRINT" ";:IF TYP$<>"CODE" THEN PRINT"    ";
250 PRINT TYP$;"  ";
260 IF TYP$<>"CODE" THEN290
270 A=FNP(B+17):E=FNP(B+19):S=FNP(B+21)
280 PRINT A;E-A;S;
290 PRINT:GOTO160
300 X= VPEEK(B+I): IF X=0 THEN X=32
310 PRINT CHR$(X);:I=I+1:RETURN
```

Tonausgabe mit dem KC 85/2(/3)

Als Leser der Zeitschrift MP möchte ich für die Rubrik MP-Computerclub Erfahrungen über die Tonausgabe mit dem KC 85/2(/3) darlegen. An der Artur-Bekker-Oberschule besteht seit etwa 2 Jahren eine Arbeitsgemeinschaft Informatik. Ziel unserer Überlegungen war es, die zweikanalige Tonausgabe des KC 85 ohne aufwendige Verstärkeranlagen zu realisieren. Mit dem SOUND-Befehl ist es beim KC 85 möglich, Töne über zwei Kanäle in einem Tonhöhenumfang von 7 Oktaven in 32 verschiedenen Lautstärken auszugeben /1/.
Technisch kann das monophon über den FBAS- oder RGB-Anschluß eines Fernsehgerätes oder über den Tonbandanschluß TAPE des Computers realisiert werden.
Um die Tonwiedergabe zweikanalig auszugeben, wurde von uns ein 2×6 W-Verstärker (Bausatz vom VEB Kontaktbauelemente und Spezialmaschinenbau Gornsdorf) aufgebaut, der im Handel als Bausatz mit fertiger Leiterplatte, Bauelementen und ausführlicher Beschreibung erworben wurde. Mit einem Diodenkabel wurde der Verstärker entsprechend Bild 6 des Systemhandbuchs S. 44 /2/ angeschlossen. An den beiden Anschlüssen 1 und 4 der TAPE-Buchse liegen je eine Ausgangsspannung von 100 mV an (Anschluß 1 – linker Kanal, Anschluß 2 – rechter Kanal). Diese Spannung reicht aus, um den Stereoverstärker mit 2 IS A 210 K im Frequenzbereich von etwa 16 Hz bzw. 25 Hz bis etwa 20 kHz auszusteuern /3/.
Der Klirrfaktor liegt bei 3 W bei etwa 0,3 %, was den Anforderungen an die AG-Arbeit sehr gut entspricht. Der Eingangswiderstand des Verstärkers liegt bei $R_E \geqq 100$ kOhm. Mit einem Tandempotentiometer von 470 kOhm kann die Lautstärke in ausreichendem Maße geregelt werden.
Der Vorteil dieses Stereoverstärkers gegenüber der monophonen Wiedergabe wurde bei vielen Beispielprogrammen deutlich. *E. Zeng*

Literatur
/1/ BASIC-Handbuch KC 85/3, VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen
/2/ Systemhandbuch KC 85/3, VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen
/3/ Bauanleitung 2×6 W-Verstärker, VEB Kontaktbauelemente und Spezialmaschinenbau Gornsdorf, Auerbacher Str., Gornsdorf, 9163

✉ KONTAKT ☎
Artur-Becker-Oberschule „Dorla",
Oberdorla, 5707,
Tel. Mühlhausen 39 19 und 24 53

ISSN 0233-2892
Mikroprozessortechnik, Berlin 1 (1987) 12
S. 353–384

COLORGRAF 0512 A3/A4

TYPY PÍSMA

AaAaAaAaAaAaAaAaAaAa

AaAaAaAaAaAaAaAaAaAaAaAaAaAaAa

AaAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaAa

AaAaAaAaAaAaAaAaAaAaAaAaAaAaAaAa

AaAaAaAaAaAaAaAaAaAaAaAaAaAa

$E=mc^2$

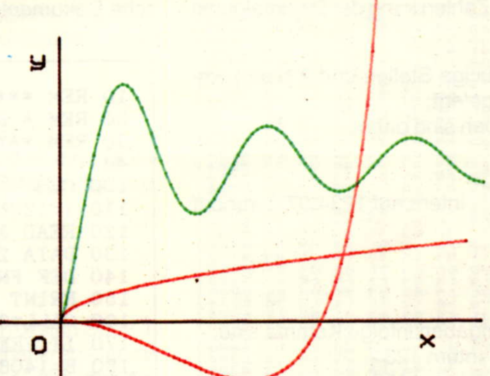
abcdefghijklmnopqrstuvwxyz.,!?+-:#*'0123456789ABCDEF

abcdefghijklmnopqrstuvwxyz.,!?+-:#*'012

abcdefghijklmnopqrstuvwxyz.,!?+

vyrábí k.p. ARITMA PRAHA

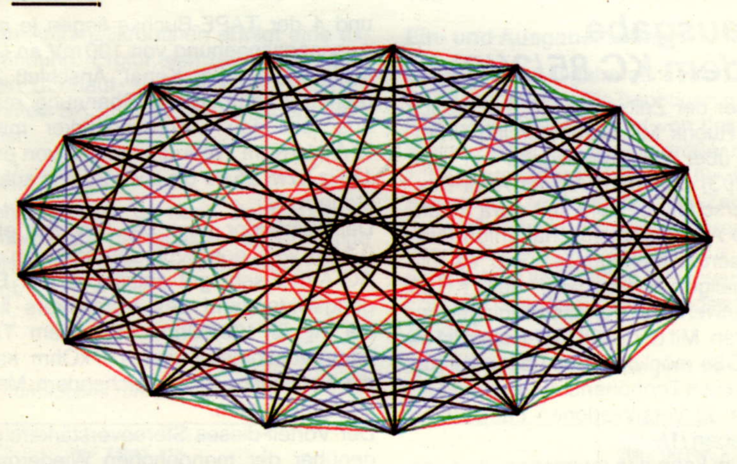
GRAF FUNKCE



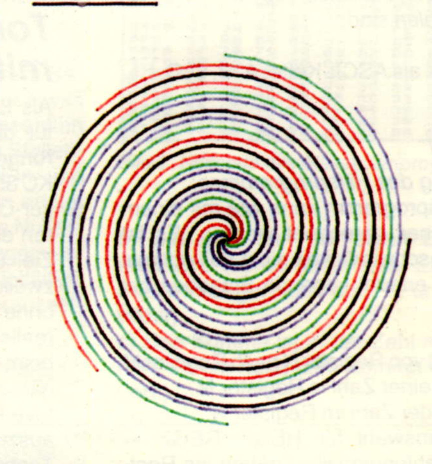
PSANÍ TEXTU

Tento předváděcí program je snahou o ukázku základních funkčních možností grafického výstupního zařízení COLORGRAF 0512, které bylo vyvinuto v k.p. ARITMA Praha na základě zkušeností s MINIGRAFem 0507.

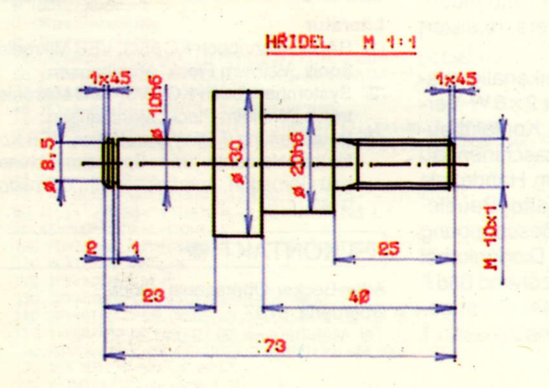
DEČKA



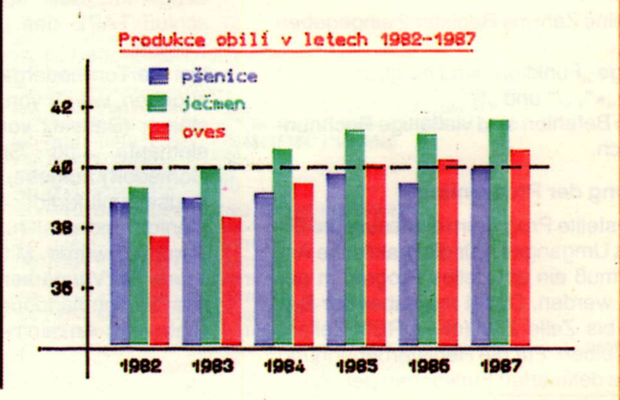
SPIRÁLA



TECHNICKÝ VÝKRES



SLOUPCOVÝ DIAGRAM



Gezeichnet auf dem COLORGRAF 0512 während der Internationalen Maschinenbaumesse Brno 1987. Zeichenzeit etwa 20 Minuten. Lesen Sie dazu bitte auch unseren Messebericht in diesem Heft.